MÁSTER EN TRATAMIENTO ESTADÍSTICO COMPUTACIONAL DE LA INFORMACIÓN

TRABAJO DE FIN DE MÁSTER

Curso 2021/2022

TRATAMIENTO DE VARIABLES CATEGÓRICAS EN MODELOS DE MACHINE LEARNING

UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS
Y
UNIVERSIDAD POLITÉCNICA DE MADRID
ESCUELA DE TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

Autor:
Rodrigo Kraus Barragán

Tutor:
Carlos Gregorio Rodríguez

Resumen

Este Trabajo de Fin de Máster tiene como principal propósito estudiar diferentes formas de codificar variables categóricas diferenciando entre ordinales y nominales, mostrando la teoría que hay detrás de cada método, detallando las ventajas e inconvenientes de estos y en qué situaciones es conveniente un codificador u otro.

Se hará también una clara distinción entre métodos clásicos para tratar estas variables y codificadores supervisados, los cuales se apoyan en la variable a predecir para sustituir cada categoría por un valor que represente la influencia que tienen sobre esta.

Se utilizará además un conjunto de datos real para apoyar la teoría con ejemplos y finalmente se aplicará lo estudiado sobre este conjunto de datos y poder comprobar qué tal funcionan los diferentes codificadores sobre estos datos con varios modelos distintos.

Palabras clave

Variables Categóricas, Minería de Datos, Machine Learning, Codificadores Clásicos, Codificadores Supervisados.

Abstract

The aim of this Master´s Thesis is to study differents ways to encoder categorical variables, including ordinal and nominal variables. Different encoders are studied, showing the advantages and disadvantages and in wich situations it is appropiate to us each of them.

It is made a distinction between classic and supervised encoders, this type of encoders replace the feature value with the influence it has over the target variable.

It is used a real dataset for putting examples of each encoder and finally the study is applicated to this dataset, showing wich encoder is better for this case.

Key words

Categorical Variables, Data Mining, Machine Learning, Classic Encoders, Supervised Encoders.

Índice

1. Introducción

Hoy en día la Inteligencia Artificial, Machine Learning, Minería de Datos, Aprendizaje Profundo o Deep Learning, etc. son temas muy nombrados de los que se escuchan numerosas noticias y aplicaciones en distintas situaciones y campos.

Se tratan de unas áreas de estudio muy amplias que están relacionadas unas con otras, incluso se puede tratar unas como subconjunto de otras. Estos temas se encuentran en evolución y en continuo estudio. En concreto, en la Minería de Datos, se puede definir como un campo multidisciplinar en el que intervienen procesos de selección, exploración y modelización de grandes cantidades de datos con el objetivo de descubrir patrones de comportamiento y finalmente adquirir conocimiento de ellos. Este conocimiento se puede llevar a una infinidad de problemas y ramas de estudio totalmente diferentes, desde el negocio en empresas con el comportamiento de clientes, campañas de marketing, etc, hasta temas de medicina como la detección de cáncer de mama o apoyo a los médicos para la toma de decisiones. La Minería de Datos se encuentra relacionada con otras áreas como el Machine Learning, las matemáticas dónde en gran parte interviene la estadística, la computación, la Inteligencia Artificial o la visualización de datos y es importante no prescindir de ninguno de estos conocimientos para poder aprender y entender bien este campo.

Como se puede observar, todo gira en torno a los datos, estos pueden proceder de diversas fuentes y lo normal es que sean de tipos diferentes. Los modelos que se emplean, en su gran mayoría, precisan que estos datos sean numéricos, sin embargo esto no es siempre así, y pueden contener variables categóricas, como se verá en la sección 2. Por tanto, es necesario saber cómo abordar este problema y no prescindir de los datos que no son numéricos porque en tal caso se estaría perdiendo información que puede llegar a ser muy valiosa.

Al igual que en otras situaciones de la Minería de Datos, no existe un método universal para codificar las variables categóricas por lo que se detallarán posibles opciones en la siguiente sección, explicando cuándo pueden ser útiles y cuándo es mejor no utilizar un método, tanto por la propia estructura del problema, como por los tipos de datos de los que se dispone o por el posible modelo a utilizar para obtener la información deseada. También se medirán aspectos como el tiempo de ejecución y la memoria consumida ya que estos aspectos pueden ser determinantes a tener en cuenta si el conjunto de datos es de gran tamaño. Se finalizará este trabajo aplicando lo visto a un caso real.

2. Métodos de Encoding de Variables Categóricas

Como se ha contado en la introducción, existen diferentes métodos para codificar las variables categóricas y poder utilizarlas en los modelos. En función de cuál sea este, puede ser interesante utilizar un método u otro, sin embargo, también puede afectar cómo es la propia variable o incluso cómo es el conjunto de datos del que se dispone. Se irá viendo y desarrollando todo esto a la par que se presentan diferentes alternativas de encoding de variables categóricas. Estudiando los distintos métodos se pueden estructurar en dos grandes grupos, codificadores clásicos y supervisados.

Primero se van a definir los distintos tipos de variables categóricas que se pueden encontrar, ya que dependiendo de cuál sea, habrá codificadores que serán más útiles que otros. Las variables categóricas pueden ser de dos tipos:

- **Nominales:** Se trata de símbolos o nombres que representan algún tipo, código o estado. Ejemplos de variable categóricas nominales son: tipos de animales o deportes, países, marcas de ropa, etc.
- **Ordinales:** Se trata de atributos que, como su propio nombre indica, presentan un posible orden. Ejemplos de este tipo de variable son: niveles de educación, sensación térmica o el grado de satisfacción de un usuario con un servicio.

A continuación se va a establecer formalmente el marco teórico para poder usar esta notación a lo largo de este trabajo. Se supone que se tiene un conjunto de datos $\mathbb{X}$ de la siguiente forma:

$$\begin{aligned} \mathbb{X} &= [\mathbf{X}; Y] \\ &= [X_1, X_2, ..., X_j, ..., X_m; Y] \end{aligned} \tag{2.1}$$

donde $\mathbf{X} = X_1, X_2, ..., X_j, ..., X_m$ es el conjunto de variables de los datos, donde cada una de las variables $X_j = x_1^j, ..., x_i^j, ..., x_n^j \ \forall j = 1, ..., m$ toma n valores, es decir, n es el número de observaciones del conjunto de datos $\mathbb{X}$. Además sin pérdida de generalidad, se supondrá que al menos la variable X_j es categórica, formada por k categorías, $[\cdot]_1, ..., [\cdot]_k$ donde se denotará por $[x_i^j]$ a la categoría de la variable X_j que pertenece la observación x_i. Por otra parte, la variable $Y = y_1, ..., y_n$ puede no estar presente por tratarse de un problema de tipo no supervisado, en caso contrario, estará reservada a la variable a predecir, si es de tipo continua, o a clasificar. En esta última situación, se tendrá que $y_i \in \{0, 1, ..., t\} \ \forall i = 1, ..., n$ donde $t \geq 2$ es el número clases. Si $t = 2$, $y_i \in \{0, 1\}$ y por tanto será un problema de clasificación binaria. De forma general también se referirá a ella como variable target o variable objetivo.

Además para poder describir algunas características de los codificadores, se va a establecer la división del conjunto de datos $\mathbf{X}$ en conjunto de entrenamiento $\mathbf{X_{TR}}$ y conjunto test $\mathbf{X_{TS}}$. En cuanto al conjunto de datos que se va a utilizar, por protección de datos estos se encuentran ofuscados. Contiene 92317 observaciones y 31 variables que son las siguientes:

- Variables $C0$, $C1$, $C2$, $C3$ y $C4$: Son cinco variables categóricas nominales que tienen 3, 9, 38, 110 y 5 categorías respectivamente.
- Variables $N00$, $N01$, ... , $N11$: Se trata de doce variables numéricas.
- Variables $O1$ y $O2$: Son dos variables categóricas ordinales.
- *País Familiar*, *País Nacimiento*, *País Nacionalidad*, *País Residencia Comunidad Autónoma Familiar*, *Comunidad Autónoma Residencia*, *Provincia Familiar*, *Provincia Nacimiento*, *Provincia Residencia*, *Municipio Familiar* y *Municipio Residencia*: Se trata de variables de tipo geográfico que representan lo que su propio nombre indica.
- Variable *Objetivo*: Se trata de la variable target, la cual es binaria por lo que el problema es de clasificación binaria.

Además este conjunto de datos hace un uso de 21,1 MB de memoria.

Por último, antes de empezar a describir los diferentes codificadores de variables categóricas, se define qué se entiende por codificador.

Definición 2.0.1 (Codificador de variable categórica)**.** *Un codificador es una aplicación,*

$$\begin{aligned} \varphi : X_j &\longrightarrow \mathbb{R}^d \\ x_i^j &\longmapsto \varphi(x_i^j) \quad \forall i = 1, ..., n \end{aligned} \tag{2.2}$$

Observación 2.0.1. *Un detalle a tener en cuenta es que la imagen de la aplicación no necesariamente tiene que estar en una dimensión, por lo que la transformación de los valores X_j podrán generarse en más de una variable nueva.*

Una vez establecido todo el marco teórico se puede comenzar a definir y detallar cada método para tratar las variables categóricas del conjunto de datos $\mathbb{X}$.

2.1. Codificadores Clásicos

Dentro de esta sección se encuentran algunos de los codificadores más conocidos y utilizados pese a que, como se verá más adelante, en ocasiones no es recomendable su utilización. Se irán presentando y describiendo los métodos que se encuentran en este grupo.

2.1.1. Ordinal Encoder

Este método es el que quizá surge de forma más natural, consiste en reemplazar cada una de las k categorías de X_j por un número entero. De esta forma la nueva variable estará formada por números enteros pertenecientes al intervalo $[1, k]$.

Definición 2.1.1 (Ordinal Encoder)**.** *El Ordinal Encoder es una aplicación,*

$$\begin{aligned} \varphi_{OE} : X_j &\longrightarrow \mathbb{N} \\ x_i^j &\longmapsto \varphi_{OE}(x_i^j) \in \{1, ..., k\} \quad \forall i = 1, ..., n \end{aligned} \tag{2.3}$$

Donde cada categoría queda representada por un y solo un número entero perteneciente al intervalo $[1, k]$*.*

Observación 2.1.1. *Un aspecto a tener en cuenta de esta codificación es que al realizar esta sustitución se le está asignando un orden a la categoría.*

Por tanto, si X_j es nominal este método no es recomendable ya que se le asignaría una ordenación que no posee y si X_j es ordinal, dicha sustitución será conveniente no hacerla de forma aleatoria sino aprovechar esta ordenación de la variable para asignar los números enteros de forma ascendente a cada categoría de la variable.

Observación 2.1.2. *Utilizar este codificador si se conocen exactamente las k categorías que toma la variable X_j.*

Como el modelo aprende sobre el conjunto de entrenamiento $\mathbf{X_{TR}}$ qué enteros asignar a cada una de las k categorías de X_j, en el caso de encontrarse con nuevas categorías en el conjunto de test $\mathbf{X_{TS}}$ o en un nuevo conjunto de predicción, no podrá codificarlas.

Aplicado al conjunto de datos que se empleará, para las variables ordinales $O1$ y $O2$ tiene sentido utilizar este método, sin embargo, si se emplea en las variables de tipo geográfico, por ejemplo a *Comunidad Autónoma Familiar* al tener veinte posibles categorías se asignaría un número entero del 1 al 20 a cada uno de ellas.

	Comunidad Autónoma	Ordinal Encoder
1	MADRID	1
2	LA RIOJA	2
3	MADRID	1
4	MADRID	1
5	MADRID	1
6	MADRID	1
7	ANDALUCIA	3
8	CASTILLA LA MANCHA	4
9	MURCIA	5
⋮	⋮	⋮

Tabla 1: Variable codificada mediante Ordinal Encoder.

De esta forma aunque se pueda pensar que tiene sentido si se asigna números altos a las comunidades autónomas más importes podría funcionar, sin embargo ¿cómo decidir qué comunidad autónoma es más importante que otra? Además, diferiría una de la anterior en una unidad, siendo esto poco razonable.

Por último, midiendo el tiempo de ejecución y la memoria consumida por este codificador al transformar el conjunto de datos, se obtienen unos resultados de 1,180747 segundos y de 21,1 MB de memoria usada. Se puede comprobar que se trata de un codificador cuya principal ventaja es la de ser rápido, añadiendo que no aumenta el uso de memoria del conjunto de datos después de codificar las variables categóricas.

2.1.2. One Hot Encoder y Dummy Encoder

Estas dos formas de tratar las variables categóricas son la más extendidas y utilizadas. Muchas veces se usan como método universal, sin pararse a pensar si es adecuada su utilización o en las posibles consecuencias de aplicar este encoder. Además, en ocasiones estos dos métodos se confunden y se tiende a pensar que es el mismo por su gran similitud pese a que se diferencian en un detalle a tener en cuenta por sus implicaciones. Ambos codificadores se basan en la idea de generar variables auxiliares binarias de forma que identifiquen de manera única las diferentes k categorías de la variable X_j.

Definición 2.1.2 (One Hot Encoder)**.** *El One Hot Encoder es una aplicación,*

$$\begin{aligned} \varphi_{OH} : X_j &\longrightarrow \mathbb{Z}_2^k \\ x_i^j &\longmapsto \varphi_{OH}(x_i^j) = (0, 0, ..., 1, ..., 0) \quad \forall i = 1, ..., n \end{aligned} \tag{2.4}$$

Donde el 1 se encontrará en la posición de la variable asignada a cada categoría.

Definición 2.1.3 (Dummy Encoder)**.** *El Dummy Encoder es una aplicación,*

$$\begin{aligned} \varphi_{DE} : X_j &\longrightarrow \mathbb{Z}_2^{k-1} \\ x_i^j &\longmapsto \varphi_{DE}(x_i^j) = (0, 0, ..., 1, ..., 0) \quad \forall i = 1, ..., n \end{aligned} \tag{2.5}$$

Donde el 1 se encontrará en la posición de la variable asignada a cada categoría.

La diferencia radica en el número de variables diferentes a generar, mientras que One Hot Encoding generaría k variables, la codificación Dummy crearía $k-1$ categorías. La razón por la que realmente no es necesaria una variable por categoría es que se podría inferir la restante ya que sería la correspondiente a no ser ninguna de las $k-1$ categorías. Sin embargo existe otra razón de peso según el modelo a utilizar por la que decantarse por la codificación Dummy Encoder, si se pretenden utilizar modelos que precisen de la inversión de la matriz de los datos, como por ejemplo una regresión lineal, si el modelo incluye un término independiente y se crea una variable para cada categoría esto puede producir que la matriz sea singular y por tanto no se pueda invertir.

En estos casos conviene no utilizar el One Hot Encoding pero esto no aplica a otros modelos que no sean sensibles a las dependencias lineales. Además, por ejemplo para los modelos basados en árboles aunque en principio pueden trabajar con variables categóricas dependiendo de la implementación esto puede no ser así y por tanto para una mayor interpretabilidad de las ramificaciones del árbol conviene utilizar el codificador One Hot y disponer explícitamente todas las categorías.

Observación 2.1.3. *Utilizar este codificador si el número de categorías k de X_j no es muy elevado.*

Cabe señalar que mediante estas codificaciones, en el caso de que X_j sea ordinal se pierde el orden que la variable poseía inicialmente. Debido a que con estos métodos no se pierde información, tienen un principal inconveniente, este es que aumenta significativamente la dimensión del problema, por ello si el número de categorías de X_j es muy elevado, no es recomendable utilizar estos codificadores. Otro problema a tener en cuenta es que, tratando las variables categóricas de esta forma, la distancia entre las categorías será únicamente de un bit.

Usando como ejemplo la variable categórica $C0$, al tener 3 categorías, se necesitarán 3 categorías (C0_1, C0_2, C0_2) en el caso del One Hot Encoder (2 usando Dummy Encoder) donde el $(1, 0, 0)$ corresponderá a la categoría L0, el $(0, 1, 0)$ a L1 y $(0, 0, 1)$ a L2.

	C0	**C0_1**	**C0_2**	**C0_3**
1	L0	1	0	0
2	L1	0	1	0
3	L1	0	1	0
4	L0	1	0	0
5	L1	0	1	0
6	L0	1	0	0
7	L0	1	0	0
⋮	⋮	⋮	⋮	⋮

Tabla 2: Ejemplo de Variable codificada mediante One Hot Encoder.

Aplicando esto al conjunto de datos del estudio, al tener numerosas variables categóricas, incluyendo las variables de tipo geográfico, de las cuales alguna como por ejemplo $C3$ tiene muchas categorías, la dimensionalidad del problema se disparará. De tener 30 variables predictoras, al aplicar el One Hot Encoder se pasa a tener 8048 siendo una cifra bastante elevada. Esto ya se refleja en el tiempo de ejecución del codificador, que eliminando las variables *Municipio Familiar* y *Municipio Residencia* es de 30,152168 segundos, y en la memoria usada del conjunto de datos una vez codificada las variables categóricas, siendo esta 2,4 GB.

2.1.3. Binary Encoder y Base-N Encoder

Son métodos que se encuentran en un punto intermedio entre los codificadores Ordinal y One Hot. Binary Encoder es un caso particular del codificador Base-N, tomando N = 2. El fin que persigue es corregir el problema de la dimensionalidad que presenta One Hot Encoder. Para ello se asigna un número entero a cada categoría de la variable X_j al igual que con el Ordinal Encoder, pero dando un paso más que consiste en pasar cada entero a la base n-ésima, de esta forma se crea para cada cifra una nueva columna.

Definición 2.1.4 (Binary Encoder)**.** *El Binary Encoder es una aplicación,*

$$\begin{aligned} \varphi_{BE} : X_j &\longrightarrow \mathbb{Z}_2^b \\ x_i^j &\longmapsto \varphi_{BE}(x_i^j) \quad \forall i = 1, ..., n \end{aligned} \tag{2.6}$$

Donde $b = \min\{b \in \mathbb{N} | b \geq \log_2(k)\}$.

Definición 2.1.5 (Base-N Encoder)**.** *El Base-N Encoder es una aplicación,*

$$\begin{aligned} \varphi_{BNE} : X_j &\longrightarrow \mathbb{Z}_2^r \\ x_i^j &\longmapsto \varphi_{BNE}(x_i^j) \quad \forall i = 1, ..., n \end{aligned} \tag{2.7}$$

Donde $r = \text{mín}\{r \in \mathbb{N} | r \geq \log_N(k)\}$.

Cabe señalar que en cierto modo aunque se formen nuevas variables, al comenzar tratando a X_j mediante el codificador Ordinal, las categorías van a presentar un orden.

Si se toma como ejemplo la variable *Comunidad Autónoma Familiar*, mientras que con el codificador One Hot eran necesarias 20 variables, con Binary Encoder se reduce a $r = \text{mín}\{r \in \mathbb{N} | r \geq \log_2(20)\} = 5$ variables.

	Comunidad Autónoma Familiar	**Comunidad Autónoma Familiar_0**	**Comunidad Autónoma Familiar_1**	**Comunidad Autónoma Familiar_2**	**Comunidad Autónoma Familiar_3**	**Comunidad Autónoma Familiar_4**
1	MADRID	0	0	0	0	1
2	LA RIOJA	0	0	0	1	0
3	MADRID	0	0	0	0	1
4	MADRID	0	0	0	0	1
5	MADRID	0	0	0	0	1
6	MADRID	0	0	0	0	1
7	ANDALUCIA	0	0	0	1	1
8	CASTILLA LA MANCHA	0	0	1	0	0
9	MURCIA	0	0	1	0	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Tabla 3: Ejemplo de Variable codificada mediante Binary Encoder.

Tras aplicar este codificador al conjunto de datos real, se tienen un total de 119 variables, frente a las 8048 del One Hot Encoder y un total de 83,8 MB de memoria utilizada. Se puede observar que efectivamente mitiga significativamente el problema de la dimensionalidad que presenta el codificador One Hot. En cuanto al tiempo de ejecución de este codificador, es de 3,194987 segundos siendo también bastante inferior.

2.1.4. Hashing Encoder

Este método se basa en la utilización de funciones hash, estas funciones son muy utilizadas en el ámbito de la criptografía. Una función hash es una función que transforma unos valores de entrada en unos valores de salida con una longitud fija denominados hashes, en criptografía, tanto la imagen como el dominio suele ser una cadena de caracteres, sin embargo para el propósito de este trabajo, la imagen de la función hash serán números enteros. Existen muchas funciones hash como por ejemplo MD5, SHA o BLAKE, para ver en más detalle consultar [8].

Definición 2.1.6 (Hashing Encoder)**.** *El Hashing Encoder es una aplicación,*

$$\begin{aligned} \varphi_{HE} : X_j &\longrightarrow \mathbb{N}^{c_0} \\ x_i^j &\longmapsto \varphi_{HE}(x_i^j) \quad \forall i = 1, ..., n \end{aligned} \tag{2.8}$$

Donde c_0 es un parámetro a elegir.

Generalmente se toma como c_0 valores que son potencias de dos. Estas funciones tienen las siguientes propiedades:

1. Únicamente se necesita el valor de entrada y de salida.
2. Son funciones deterministas y de una sola dirección.
3. Se pueden producir colisiones entre los hashes

Es por la propiedad 1 que esté método de codificación de variables categóricas es poco costoso en cuanto a la memoria que consume. Como se describe en la propiedad 2, son funciones deterministas, lo cual es muy importante ya que de no ser así una categoría, tras aplicar sobre ella la función hash, podría derivar en diferentes hashes. También se ha mencionado que son funciones unidireccionales, por lo que una vez aplicado la función hash no hay forma de deshacer la operación. Si la variable X_j tiene un número k de categorías finito y conocido, entonces sí se puede realizar una tabla para ver cada valor de hash de qué categoría proviene, por el contrario si durante el proceso de test o de predicción se encuentran nuevas categorías, aunque este encoder sea capaz de lidiar con ello no se sabrá controlarlo. Por último, como se indica en la propiedad 3 pueden darse colisiones entre los hashes, es decir dos categorías diferentes pueden pueden tener el mismo valor de hash, a menor c_0 existe una mayor riesgo de que se produzcan colisiones.

Gracias a estas funciones hash, se pueden codificar las variables categóricas con un gran número de categorías en unas nuevas variables que recojan estos hashes. En gran medida, la presencia de colisiones dependerá del número de nuevas categorías que se escojan.

A modo de resumen, este método puede ser útil cuando se tienen muchas variables categóricas o cuando estas variables tienen muchos tipos diferentes de categorías, por ejemplo una variable que recoja el nombre de diferentes calles.

Empleando este codificador sobre las variables categóricas nominales del conjunto de datos del estudio con un valor de $c_0 = 8$ y utilizando el método MD5 se invierte 64,481811 segundos y un consumo de 15,5 MB en memoria.

	Col 0	**Col 1**	**Col 2**	**Col 3**	**Col 4**	**Col 5**	**Col 6**	**Col 7**
1	0	4	0	7	0	0	5	0
2	3	4	1	3	0	0	5	0
3	0	4	2	7	0	0	3	0
4	0	2	1	6	2	0	5	0
5	0	4	2	7	0	0	3	0
6	0	4	1	4	0	1	6	0
7	0	4	3	0	2	0	4	3
8	0	4	1	3	2	0	6	0
9	0	4	1	3	0	4	4	0
10	2	4	2	3	0	1	4	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Tabla 4: Ejemplo donde todas las variables categóricas han sido codificadas mediante Hashing Encoder.

Mediante este encoder se sintetiza toda la información en esas c_0 variables por lo que la dimensionalidad no solo puede no aumentar, sino que se puede reducir respecto a la original. En cambio, esto requiere un coste computacional en el tiempo que invierte en codificar las variables categóricas.

2.1.5. Count Encoder y Frequency Encoder

Tanto Count Encoder como Frequency Encoder son métodos para codificar las variables categóricas que resultan muy rápidos de aplicar. Se basan en reemplazar cada una de las k categorías de X_j por su frecuencia absoluta o relativa respectivamente, es decir, Count Encoder reemplaza cada categoría por las veces que aparece en el conjunto de datos mientras que Frequency Encoder da un paso más y se divide por el número de observaciones.

Definición 2.1.7 (Count Encoder)**.** *El Count Encoder es una aplicación,*

$$\begin{aligned} \varphi_{CE} : X_j &\longrightarrow \mathbb{N} \\ x_i^j &\longmapsto \varphi_{CE}(x_i^j) = n_{[x_i^j]} \quad \forall i = 1, ..., n \end{aligned} \tag{2.9}$$

Donde $n_{[x_i^j]}$ es el número de veces que aparece la categoría de x_i en X_j.

Definición 2.1.8 (Frequency Encoder)**.** *El Frequency Encoder es una aplicación,*

$$\begin{aligned} \varphi_{FE} : X_j &\longrightarrow (0, 1) \\ x_i^j &\longmapsto \varphi_{FE}(x_i^j) = \frac{n_{[x_i^j]}}{n} \quad \forall i = 1, ..., n \end{aligned} \tag{2.10}$$

Donde $n_{[x_i^j]}$ es el número de veces que aparece la categoría de x_i en X_j.

Observación 2.1.4. *La codificación por frecuencia quedará en el intervalo abierto $(0, 1)$. Cero no puede ser porque se encuentra mínimo una vez y tampoco 1 porque en dicho caso significaría que la variable X_j únicamente tiene una categoría y por tanto debería excluirse del modelo ya que no aportaría ninguna información.*

Observación 2.1.5. *La estimación de $n_{[x_i^j]}$ deberá realizarse sobre el conjunto de entrenamiento $\mathbf{X_{TR}}$.*

Este último tiene la ventaja de que los datos ya se encontrarían normalizados, es decir, los valores estarían en el intervalo $(0, 1)$, esto hará que a la hora de entrenar un modelo y a la hora de realizar un predicción, se tenga en cuenta el tamaño de la muestra, de no ser así puede haber mucha diferencia entre los valores sobre los que el modelo ha aprendido durante el entrenamiento y entre los valores de la muestra sobre la que se quiere predecir o por ejemplo sobre el conjunto de test.

Estos métodos pueden funcionar bien en el caso de encontrarse en un problema de aprendizaje supervisado y la frecuencia de las variables categóricas tengan relación con la variable respuesta, ya que puede ayudar los modelos a aprender durante el entrenamiento el peso que dar a X_j. Sin embargo tiene un principal inconveniente, si alguna de las k categorías aparece el mismo número de veces entonces estas se codificarán con el mismo valor, por lo que no habrá distinción entre ellas.

Empleando Frequency Encoder por ejemplo a la variable *Comunidad Autónoma Familiar* se puede observar que la categoría *MADRID* aparece bastante más que por ejemplo *LA RIOJA*:

	Comunidad Autónoma Familiar	**Frequency Encoder**
1	MADRID	0,723399
2	LA RIOJA	0,004550
3	MADRID	0,723399
4	MADRID	0,723399
5	MADRID	0,723399
6	MADRID	0,723399
7	ANDALUCIA	0,029572
⋮	⋮	⋮

Tabla 5: Variable codificada mediante Frequency Encoder.

Los resultados que se obtienen al aplicarlo sobre las variables categóricas del conjunto de datos se emplean 2,599651 segundos y 21,1 MB de memoria, la misma que la orignal. Se trata de un codificador también rápido.

2.2. Codificadores Supervisados

En esta sección se describirán unos métodos un tanto diferentes para tratar las variables categóricas. A diferencia de los contrastes clásicos que se basan en operaciones sobre esa misma variable, este grupo hace uso de la variable target u objetivo, de ahí el nombre de codificadores supervisados, para reemplazar cada categoría de X_j por un valor que refleje los efectos que tiene esta sobre la variable respuesta Y. Es por ello que si el problema requiere un modelo de aprendizaje no supervisado, estos métodos no se podrán utilizar. También cabe destacar que, pese a utilizar la variable a predecir o clasificar, no se tiene porqué obtener mejores resultados que utilizando los codificadores de la sección anterior.

2.2.1. Target Encoder

La idea que hay detrás de este método es utilizar las probabilidades condicionadas de la variable respuesta a cada una de las categorías. Sin embargo, por motivos que se explicarán más tarde, no se sustituye únicamente por este valor, sino que en el caso de tratarse de un problema de clasificación, cada categoría se reemplaza por una combinación entre la probabilidad a posteriori de la variable respuesta dada esa categoría y la probabilidad a priori de la variable respuesta. En cambio, si se trata de un problema de regresión se sustituirá por la combinación del valor esperado de la variable respuesta dada cada categoría y el valor esperado de la variable respuesta. Siguiendo la línea de desarrollo del artículo [15], se explicará el caso de clasificación binaria y se extenderá la teoría para los problemas clasificación multiclase y regresión.

Al tratarse de un problema de clasificación binaria, $Y \in \{0, 1\}$ como se comentó al principio se puede asignar a cada categoría de X_j la probabilidad condicionada de la variable respuesta a cada una de las categorías, es decir, $x_i \longmapsto P(Y = 1|X_j = [x_i^j])$. La pregunta ahora es cómo estimar esta probabilidad.

Lo primero a tener en cuenta es que las estimaciones deben realizarse sobre el conjunto de entrenamiento $\mathbf{X_{TR}}$. Si el tamaño de la muestra de $\mathbf{X_{TR}}$ es suficientemente grande una estimación razonable sería:

$$x_i \longmapsto \frac{n_{[x_i^j],Y=1}}{n_{[x_i^j]}} \quad \forall i = 1, ..., n_{TR} \tag{2.11}$$

Donde $n_{[x_i^j],Y=1}$ es el número de veces que aparece la categoría de x_i en la variable X_j y que la variable binaria es igual a 1.

Si se finzaliza en este paso es lo que se conoce como Mean Encoder, sin embargo, el problema surge cuando o bien la muestra de entrenamiento $\mathbf{X_{TR}}$ es pequeña o cuando alguna de las categorías de X_j tiene una cardinalidad muy baja porque entonces la estimación de $P(Y = 1|X_j = [x_i^j])$ sugerida en 2.11 no será muy fiable. Es por esto que Daniele Micci-Barreca en [15] propone una estimación alternativa que se trata de una una combinación entre $P(Y = 1|X_j = [x_i^j])$ y $P(Y = 1)$. Es decir, propone el siguiente encoder que es lo que se conoce como Target Encoder:

Definición 2.2.1 (Target Encoder)**.** *El Target Encoder para un problema de clasificación binaria es una aplicación:*

$$\begin{aligned} &\varphi_{TE} : X_j \longrightarrow [0,1] \\ &\qquad x_i^j \longmapsto \varphi_{TE}(x_i^j) = \lambda(n_{[x_i^j]})\frac{n_{[x_i^j],Y=1}}{n_{[x_i^j]}} + (1-\lambda(n_{[x_i^j]})\frac{n_{Y=1}}{n_{TR}} \quad \forall i = 1, ..., n_{TR} \end{aligned} \tag{2.12}$$

Donde:

- $n_{[x_i^j],Y=1}$ *es el número de observaciones de la categoría de* x_i *en* X_j *con* $Y = 1$*.*
- $n_{[x_i^j]}$ *es el número de veces que aparece la categoría de* x_i *en* X_j*.*
- $n_{Y=1}$ *es el número de observaciones con la variable respuesta igual a* 1*.*
- n_{TR} *es el número total de observaciones en el conjunto de entrenamiento* $\mathbf{X_{TR}}$*.*
- $\lambda(n_{[x_i^j]})$ *es una función monótona creciente comprendida en el intervalo* $[0, 1]$ *dependiente de la muestra de la clase de* x_i *en* X_j*.*

Observación 2.2.1. *Esta función* $\lambda(n_{[x_i^j]})$ *sirve para controlar qué peso dar a la estimación de* $P(Y = 1|X_j = [x_i^j])$ *dada en 2.11.*

Cuando la función valga 1 todo el peso se le dará a esta estimación e irá reduciéndose en función de que el valor se acerque a 0, que en ese caso será equivalente a codificar la variable por la probabilidad a priori de la variable respuesta binaria Y, es decir, que conocer el valor que toma la variable X_j no proporciona ninguna información extra sobre que la variable Y sea igual a 1.

Una posible función que se propone en dicho artículo y en concreto es la que se encuentra implementada en la librería de Python Category Encoders [14], es la siguiente:

$$\lambda(a) = \frac{1}{1 + e^{-\frac{a-h}{f}}} \tag{2.13}$$

Esta función tiene forma de S y como se puede observar tiene dos hiperparámetros a seleccionar, el primero es h que establece el valor a partir del cual la función $\lambda(a)$ vale 0.5 por lo que significa que h establece la mitad del tamaño mínimo de $n_{[x_i^j]}$ para que se confíe totalmente en la estimación de 2.11, mientras que el segundo valor es f que controla la transición entre las partes de la fórmula de 2.12.

Observación 2.2.2. *Al establecer de esta forma la codificación de la variable categórica, permite también controlar y suavizar en cierta forma el posible sobreajuste que se podría cometer si únicamente se sustituyen los valores de* x_i^j *por el resultado de la ecuación 2.11.*

Todo esto se ha descrito para el caso de un problema de clasificación binaria, sin embargo es fácilmente extensible a los casos de clasificación multiclase, donde la variable de respuesta Y pueda tomar más de dos valores, o a los problemas de regresión, donde Y será continua.

Comenzando por el problema de clasificación multiclase, se supondrá que se tienen t clases diferentes, entonces, $y_i = 0, 1, ..., t$. Como toda la información de las probabilidades condicionadas de Y a cada una de las categorías no se puede condensar en una sola variable, se crearán t nuevas variables, donde en cada una de ellas quedará reflejada la combinación entre la estimación de $P(Y = y_i|X_j = [x_i^j])$ y la estimación de $P(Y = y_i)$.

En el caso de tener un número elevado t de clases, esto puede conllevar un aumento considerable de la dimensión del problema. Aunque en la mayoría de casos los problemas de clasificación no suelen tener muchas clases, es algo que hay que tener en cuenta.

Observación 2.2.3. *Al igual que en el problema de clasificación binaria no eran necesarias dos variables porque con tener una probabilidad se tiene su complementaria, en el problema de clasificación de t clases bastará con quedarse con $t-1$ variables nuevas y por tanto eliminar una cualquiera sin perder información.*

Continuando con el último de los casos, el problema de regresión, la codificación resultaría de la combinación entre $E[Y|X_j = [x_i^j]]$ y $E[Y]$. Por tanto la codificación equivalente a 2.12 para un problema de regresión sería:

$$x_i^j \longmapsto \lambda(n_{[x_i^j]})\frac{\sum\limits_{i\in C_c} y_i}{n_{[x_i^j]}} + (1 - \lambda(n_{[x_i^j]})\frac{\sum\limits_{i=1}^{n_{TR}} y_i}{n_{TR}} \quad \forall i = 1, ..., n_{TR} \tag{2.14}$$

Donde C_c $\forall c = 1, ...k$ son los conjuntos de observaciones de las diferentes k categorías de la respectiva variable categórica X_j.

Con todo esto ya quedan cubiertos y descritos todos los posibles casos que se pueden presentar y se ha descrito cómo funciona el Target Encoder.

Observación 2.2.4. *Este codificador tiene una ventaja y es que los valores que devuelve, se encuentran en el intervalo* $[0, 1]$*, lo cual es muy útil para modelos como pueden ser las Redes Neuronales.*

Observación 2.2.5. *Como se ha visto en el codificador Target Encoder, se realiza una estimación de la probabilidad a priori de* Y*, sin embargo, puede ser interesante siguiendo la estadística bayesiana, establecer su distribución a priori y no calcularla mediante los datos.*

La fórmula utilizada en el codificador Target Encoder para tratar las variables categóricas se toma de un área conocida como Empirical Bayes. Esta surge para estimar la probabilidad a priori de los datos y no establecerla como información inicial, en contra de lo que dicta la estadística bayesiana.

La ecuación general utilizada en Empirical Bayes es:

$$\hat{P}_{[x_i^j]]} = B_{[x_i^j]]}\hat{y}_{[x_i^j]]} + (1 - B_{[x_i^j]]})\hat{y} \tag{2.15}$$

Donde $\hat{y}_{[x_i^j]]}$ es la probabilidad a posteriori empírica para la categoría de x_i en la variable X_j y $\hat{y}$ es la probabilidad a priori estimada a través de los datos. Un caso particular cuando la distribuciones de los datos se asumen ser Gaussianas, es tomar como $B_{[x_i^j]]}$ la siguiente expresión:

$$B_{[x_i^j]]} = \frac{n_{[x_i^j]]}\tau^2}{\sigma^2 + n_{[x_i^j]]}\tau^2} \tag{2.16}$$

Donde τ^2 es la varianza de la muestra y σ^2 es la varianza de la muestra fijada una categoría de x_j, teniendo así unos pesos dependientes no solo del tamaño de la muestra si no de su varianza dentro de cada categoría y total.

Por último cabe mencionar que este método para codificar variables categóricas puede verse bastante afectado por la presencia de outliers en la variable Y en caso de ser continua, por lo que en la medida de lo posible conviene realizar un buen tratamiento de outliers antes de aplicar este método a la variable X_j.

Aplicando esto al conjunto de datos y mostrando como ejemplo la variable *Comunidad Autónoma Familiar*:

	Comunidad Autónoma Familiar	**Target Encoder**
1	MADRID	0,139828
2	LA RIOJA	0,111905
3	MADRID	0,139828
4	MADRID	0,139828
5	MADRID	0,139828
6	MADRID	0,139828
7	ANDALUCIA	0,160073
⋮	⋮	⋮

Tabla 6: Variable codificada mediante Target Encoder.

Como se dijo se puede observar que se obtienen valores dentro del intervalo [0,1], en concreto, valores cercanos a cero debido a que la variable *Objetivo* se encuentra desbalanceada.

Lo primero es decidir el valor de los hiperparámetros, la idea es controlar tanto h como f para dar la forma que se desee a la función λ. Los valores por defecto que se encuentran en [14] para ambos es 1, esto lo que produce es que para categorías con una sola aparición, se le dé el mismo peso a ambas probabilidades estimadas, para $n_{[x_i^j]} = 3$ se confia al 90 % y para valores mayores a 7 se confía al 99 %. Esto no parece muy razonable, por ello siguiendo la discusión sobre el tema que se puede encontrar en [24], tomando $h = 20$ y $f = 10$ se establece que si la categoría únicamente aparece una sola vez, se confíe al 95 % en la media global, para $n_{[x_i^j]} = 10$, se le dé una importancia del 75 % a la media global, la igualdad de pesos se alcanza cuando $n_{[x_i^j]} = 20$ y prácticamente todo el peso sea para $P(Y = 1|X_j = [x_i^j])$ cuando $n_{[x_i^j]} = 50$.

Este codificador empleó 2,699451 segundos por lo que se trata de un encoder bastante rápido, que además no aumenta la memoria utilizada respecto al conjunto de datos original

2.2.2. Leave One Out Encoder

El método Leave One Out (LOO) para codificar variables categóricas muy similar al codificador Mean Encoder. Surge para intentar reducir el inconveniente de la presencia de los outliers mencionado al final de la sección anterior. Para ello emplea la misma ecuación que el Mean Encoder pero, como su nombre indica, consiste en no incluir la observación x_i^j en la

fórmula 2.11 y de esta forma mitigar los efectos de los outliers.

Sin embargo si el conjunto de entrenamiento $\mathbf{X_{TR}}$ es muy grande este método puede aumentar el coste computacional.

Observación 2.2.6. *Una forma para intentar paliar el riesgo de sobreajuste tanto en el método Leave One Out como en Target Encoder, es añadir ruido gaussiano a cada observación de* $\mathbf{X_{TR}}$.

Aplicando este codificador al conjunto de datos con un ruido gaussiano con desviación típica de 0,05 y tomando como ejemplo de nuevo la variable *Comunidad Autónoma Familiar*:

	Comunidad Autónoma Familiar	**LOO Encoder**
1	MADRID	0,136631
2	LA RIOJA	0,113470
3	MADRID	0,136409
4	MADRID	0,138353
5	MADRID	0,143675
6	MADRID	0,147576
7	ANDALUCIA	0,164696
⋮	⋮	⋮

Tabla 7: Ejemplo de variable codificada mediante LOO Encoder.

Puede verse que, tanto por la propia definición del codificador como por el ruido gaussiano añadido, una misma categoría puede ser reemplazada por valores diferentes. Respecto al tiempo de ejecución, invierte 1,7034971 segundos, consumiendo la misma memoria que el dataset original.

2.2.3. CatBoost Encoder

Este codificador para tratar las variables categóricas, propuesto por los autores de los artículos [6] y [19] surge apoyándose en la idea de los algoritmos online los cuales obtienen muestras de entrenamiento de forma secuencial a lo largo del tiempo. Para adaptar esta idea se genera una permutación aleatoria de los datos y se realiza el cálculo que se verá más adelante solo con las observaciones ya encontradas. De esta forma se reduce el problema de sobreajuste del que sufren codificadores como el Target Encoder. Si únicamente se toma como guía temporal una única permutación el resultado no será muy fiable por lo que conviene realizar varias permutaciones del conjunto de datos.

Teniendo todo esto en cuenta se define el CatBoost Encoder de la siguiente forma:

Definición 2.2.2 (CatBoost Encoder)**.** *El CatBoost Encoder es una aplicación,*

$$\varphi_{CBE} : \sigma(X_j) \longrightarrow \mathbb{R}$$

$$x^j_{\sigma_p} \longmapsto \varphi_{CBE}(x^j_{\sigma_p}) = \frac{\sum_{l=1}^{p-1} \mathbb{1}\{x_{\sigma_l} = x_{\sigma_p}\} y_{\sigma_l} + a * P(Y)}{\sum_{l=1}^{p-1} \mathbb{1}\{x_{\sigma_l} = x_{\sigma_p}\} + a} \quad \forall x^j_i \in \mathbf{X_{TR}} \tag{2.17}$$

Donde:

- $\mathbb{1}\{x_{\sigma_l} = x_{\sigma_p}\}$ *vale uno si* $x_{\sigma_l} = x_{\sigma_p}$ *y 0 en otro caso.*
- y_{σ_l} *es el valor del target* Y *para esa observación.*
- $P(Y)$ *es la probabilidad a priori del target* Y
- a*, es un hiperparámetro a seleccionar que representa el peso que se le da a la probabilidad a priori.*

El porqué del nombre CatBoost Encoder es debido a la forma de codificar las variables categóricas que se realiza en el algoritmo de Gradient Boosting desarrollado por los autores mencionados anteriormente, al juntar ambas partes surge este nombre. Sin embargo, no se utiliza el modelo de Gradient Boosting orginal, sino que se trata de una variante.

Esta variante denominada Ordered Boosting trata de evitar realizar cada iteración del algoritmo sobre el mismo conjunto de datos para no caer en el sobreajuste, para ello realiza varias permutaciones y utiliza estas en cada iteración para entrenar el modelo con datos diferentes.

Este Gradient Boosting sirve tanto para problemas de clasificación como problemas de regresión, por tanto a la hora de enfrentarse a ellos, cuando se tienen variables categóricas en el conjunto de datos es importante considerar no solo este método de codificación sino también considerar esta variante de Gradient Boosting que incluye a su vez este encoder para las variables categóricas. Para consultar más en detalle este modelo se puede consultar los artículos de [6] y [19] así como la página web de la librería para llevar a cabo su implementación mencionada en [6].

Llevando este codificador a la práctica, tomando $a = 1$, sobre el conjunto de datos del estudio, y en particular sobre la variable *Comunidad Autónoma Familiar*:

	Comunidad Autónoma Familiar	**CatBoost Encoder**
1	MADRID	0,142043
2	LA RIOJA	0,142043
3	MADRID	0,071022
4	MADRID	0,047348
5	MADRID	0,035511
6	MADRID	0,028409
7	ANDALUCIA	0,142043
⋮	⋮	⋮

Tabla 8: Ejemplo de variable codificada mediante CatBoost Encoder.

Este codificador invierte 1,57431 segundos en tratar la variable categórica, manteniendo los 21,1 MB del conjunto de datos original.

2.2.4. M-estimate Encoder

Este método propuesto por Cestnik en [4] surge como método para estimar las probabilidades condicionales. Cuando se hace uso de la ecuación de Bayes para problemas de clasificación en modelos de Machine Learning, es común abordarlo asumiendo independencia entre variables y la precisión con la que se consiga clasificar depende en gran medida de la estimación que se haga de las probabilidades condicionales.

Si se tiene más de una variable categórica $X_j \ \forall j = 1, ..., l$ donde puede tomar diferentes categorías cualesquiera, se supone que cada una toma los valores $V^1, ..., V^l$. Teniendo un problema de clasificación con t clases diferentes se tendría que:

$$\begin{aligned} P(t|V^1 \cdots V^l) &= \frac{P(tV^1 \cdots V^l)}{P(V^1 \cdots V^l)} \\ &= P(t)\frac{P(t|V^1)}{P(t)}\frac{P(t|V^1V^2)}{P(t|V^1)}\frac{P(t|V^1V^2V^3)}{P(t|V^2)}\cdots \end{aligned} \tag{2.18}$$

Y como se comentó al principio, asumiendo independencia entre las categorías de las variables:

$$P(t|V^1 \cdots V^l) = P(t)\frac{P(t|V^1)}{P(t)}\frac{P(t|V^2)}{P(t)}\frac{P(t|V^3)}{P(t)} \tag{2.19}$$

Por lo que todo queda resumido a realizar una correcta estimación de $P(t|V^j)$. En lugar de suponer una distribución inicial uniforme, Cestnik propone utilizar como función de densidad inicial la siguiente:

$$f(x) = \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1} \tag{2.20}$$

donde $a, b > 0$ y $B(a,b)$ es la función beta. Por tanto, después de n aciertos en N intentos, la esperanza de la probabilidad de acierto en el siguiente intento es:

$$q(n,N) = \frac{n+a}{N+a+b} \tag{2.21}$$

Además se determinará un hiperparámetro m que se definirá de forma que $a + b = m$. Este parámetro se deberá escoger en función del ruido que se espere en el conjunto de datos, a mayor ruido mayor valor de m. Una vez establecido m se definen los parámetros a y b cumpliendo $a = P(t)m$ y $b = m - a$. De esta forma, esta aproximación cumple las siguientes propiedades que se dejarán como proposición:

Proposición 2.2.1. *La ecuación 2.21 verifica las siguientes propiedades:*

i) $q(0,0) = P(t)$

ii) $q(0,N) > 0$

iii) $q(N,N) < 1$

iv) $q(N+1,N+1) > q(N,N)$

v) $q(0,N+1) < q(0,N)$

Demostración.

- *i)* Una vez determinado $q(0,0) = \frac{a}{a+b} = \frac{P(t)m}{m} = P(t)$ □
- *ii)* $q(0,N) = \frac{a}{N+a+b} = \frac{P(t)m}{N+m} > 0$ □
- *iii)* $q(N,N) = \frac{N+a}{N+a+b} = \frac{N+P(t)m}{N+m} < \frac{N+m}{N+m} = 1$ □
- *iv)* $q(N+1,N+1) = \frac{N+1+a}{N+1+a+b} = 1 - \frac{b}{N+1+a+b} > 1 - \frac{b}{N+a+b} = \frac{N+a}{N+a+b} = q(N,N)$ □
- *v)* $q(0,N+1) = \frac{a}{N+1+a+b} < \frac{a}{N+a+b} = q(0,N)$ □

□

Una vez visto todo esto, se utilizará esta q como fórmula para codificar las variables categóricas de la siguiente forma:

Definición 2.2.3 (m-Estimate Encoder)**.** *El m-Estimate Encoder para un problema de clasificación binaria es una aplicación:*

$$\begin{aligned} &\varphi_{mE} : X_j \longrightarrow [0,1] \\ &\qquad x_i^j \longmapsto \varphi_{mE}(x_i^j) = \frac{n_{[x_i^j],Y=1} + \frac{n_{Y=1}}{n_{TR}} m}{n_{[x_i^j]} + m} \quad \forall i = 1, ..., n_{TR} \end{aligned} \tag{2.22}$$

Donde:

- $n_{[x_i^j],Y=1}$ *es el número de observaciones de la categoría de* x_i *en* X_j *con* $Y = 1$.
- $n_{[x_i^j]}$ *es el número de veces que aparece la categoría de* x_i *en* X_j.
- $n_{Y=1}$ *es el número de observaciones con la variable respuesta igual a* 1.
- n_{TR} *es el número total de observaciones en el conjunto de entrenamiento* $\mathbf{X_{TR}}$.
- $\frac{n_{Y=1}}{n_{TR}}$ *es la estimación de la a priori* $P(Y = 1)$.
- $m > 0$ *es un hiperparámetro a establecer, según Cestnik, cuanto mayor ruido tenga el conjunto de datos, mayor tiene que ser* m.

Esto para el caso de tener un problema de clasificación binaria, para la clasificación multiclase y el problema de regresión se realizará de la misma forma a como se vio en Target Encoder.

Observación 2.2.7. *Realmente Target Encoder y m-Estimate Encoder coinciden si se toma* $\lambda(n) = \frac{n}{m+n}$ *pero tiene la ventaja de solo precisar de un hiperparámetro frente a los dos del Target Encoder.*

Por último cabe mencionar que Cestnik, autor del artículo mencionado anteriormente, junto a Bratko proponen un método de poda de árboles basada en la m-probabilidad estimada [5].

Aplicando el codificador m-Estimate al conjunto de datos de estudio, fijando $m = 1$ y tomando como ejemplo la variable *Comunidad Autónoma Familiar*:

	Comunidad Autónoma Familiar	**m-Estimate Encoder**
1	MADRID	0,139828
2	LA RIOJA	0,111976
3	MADRID	0,139828
4	MADRID	0,139828
5	MADRID	0,139828
6	MADRID	0,139828
7	ANDALUCIA	0,160067
⋮	⋮	⋮

Tabla 9: Ejemplo de variable codificada mediante m-Estimate Encoder.

Esta forma de codificar las variables categóricas requirió de 2,353187 segundos, manteniendo la misma memoria que el conjunto de datos original.

2.2.5. James-Stein Encoder

Este estimador se basa en el estimador James Stein. Para explicar la idea se supone lo siguiente, un entrenador de baloncesto está realizando unas pruebas de acceso para incorporar a gente nueva en su equipo. El año pasado su equipo tuvo muy mal porcentaje en tiros libres y quiere que los nuevos fichajes tengan mejor capacidad de anotación. A las pruebas se presentan 5 candidatos a los cuales les hace tirar 10 tiros libres anotando respectivamente 8, 6, 3, 4 y 5 tiros libres. El entrenador calcula la proporción de acierto de cada uno, que respectivamente es 0,8, 0,6, 0,3, 0,4 y 0,5. El entrenador que sabe que la proporción de su equipo del año pasado fue de 0,41 decide quedarse solo con el primer, segundo y último candidato pero, ¿realmente esta estimación es buena y se acercará a la proporción de tiros libres a final de temporada? La respuesta es que no es adecuada esta estimación y existe una mejor, aquí es donde entra la idea de Stein que consiste en ponderar cada uno de los porcentajes de tiro de cada uno con el porcentaje medio de todos.

Definición 2.2.4 (Estimador James-Stein)**.** *Se define el estimador James-Stein como:*

$$JS_i = \overline{y_i} + c(\overline{y} - \overline{y_i}) \tag{2.23}$$

donde $\overline{y_i}$ es la media de la muestra inicial, $\overline{y}$ es la media del total y c es el parámetro que controla la ponderación entre ambas medias.

El parámetro c tiene la siguiente expresión, $c = \dfrac{(k-3)\sigma^2}{\sum_{i=1}^{k}(\overline{y_i} - \overline{y})^2}$ done k es el número de medias a estimar y σ^2 es la varianza.

Esta idea que parece ir en contra de lo intuitivo porque puede plantearse qué tendrá que ver la proporción de tiro de un candidato con la de otro puesto que son independientes, de ahí que se le conozca como Paradoja de Stein. Para consultar en más detalle este estimador consultar [23] y [11].

Suponiendo que los datos pertenecen a una *Normal* de desviación típica 0,15, se calcula este coeficiente c:

$$c = \frac{(5-3)0{,}01}{0{,}0784 + 0{,}0064 + 0{,}0484 + 0{,}0144 + 0{,}0004} = 0{,}135$$

Por lo que se tiene que cada estimador JS es:

1) $JS_1 = 0{,}8 + 0{,}304(0{,}52 - 0{,}8) = 0{,}715$

2) $JS_2 = 0{,}6 + 0{,}304(0{,}52 - 0{,}6) = 0{,}576$

3) $JS_3 = 0{,}3 + 0{,}304(0{,}52 - 0{,}3) = 0{,}367$

4) $JS_4 = 0{,}4 + 0{,}304(0{,}52 - 0{,}4) = 0{,}436$

5) $JS_5 = 0{,}5 + 0{,}304(0{,}52 - 0{,}5) = 0{,}506$

Mediante esta estimación también debería quedarse con el cuarto candidato.

Volviendo al estudio de codificar las variables categóricas, la forma de aplicar esta idea es:

Definición 2.2.5 (James-Stein Encoder)**.** *El James-Stein Encoder es una aplicación,*

$$\begin{aligned} \varphi_{JS} : X_j &\longrightarrow \mathbb{R} \\ x_i^j &\longmapsto \varphi_{JS}(x_i^j) = (1-B)\overline{y}_{[x_i^j]]} + B\overline{y} \quad \forall i = 1, ..., n_{TR} \end{aligned} \tag{2.24}$$

Donde:

- $\overline{y}_{[x_i^j]]}$ *es la media de la variable target para la categoría de la observación i-ésima de la variable* X_j*.*
- $\overline{y}$ *es la media de la variable* Y*.*
- $B = \dfrac{\hat{\tau}}{\hat{\tau} + \hat{\sigma^2}}$ *que respectivamente* $\hat{\tau}$ *es la varianza de* $\overline{y}_{[x_i^j]]}$ *y* $\hat{\sigma^2}$ *es la varianza de* Y*.*

Como se puede ver el parámetro B es el encargado de ponderar el peso que se le da a la media de Y para cada categoría de X_j. A mayor varianza de $\overline{y}_{[x_i^j]]}$ mayor B y por tanto, se reducirá más el peso que se le da a la media de y fijada una categoría para evitar sobreajustar el modelo al conjunto de entrenamiento.

Observación 2.2.8. *El codificador James-Stein está contruido para cuando la variable target Y sigue una distribución normal.*

A continuación se va a realizar el procedimiento desde el punto de vista bayesiano. Supóngase que la media de Y para una categoría c es $N(\theta_c, \sigma_c^2)$ donde σ_c^2 es conocida y la distribución a priori sobre θ_c es $N(\mu_0, \sigma_0^2)$. Se tiene entonces que:

- $\pi(\theta_c) = N(\mu_0, \sigma_0^2)$
- $f(x_1, ..., x_n|\theta_c) = N(\theta_c, \sigma_c^2)$

Por tanto, la función de distribución final $\pi(\theta_C|x_1, ..., x_n)$ es de la forma:

$$\pi(\theta_C|x_1, ..., x_n) = \frac{\pi(\theta_c)f(x_1, ..., x_n|\theta_c)}{\displaystyle\int_{-\infty}^{\infty} \pi(\theta_c)f(x_1, ..., x_n|\theta_c)d\theta} \tag{2.25}$$

Desarrollando la ecuación 2.25 se llega a que la distribución final sigue una $N(\mu_1, \sigma_1)$ donde:

$$\mu_1 = \frac{\dfrac{\mu_0}{\sigma_0^2} + \dfrac{n\overline{x}}{\sigma_c^2}}{\dfrac{1}{\sigma_0^2} + \dfrac{n}{\sigma_c^2}}$$

$$\sigma_1 = \frac{1}{\sqrt{\dfrac{1}{\sigma_0^2} + \dfrac{n}{\sigma_c^2}}}$$

Una vez obtenida esto, si se multiplica el numerador y denominador de μ_1 por $\frac{\sigma_0^2\sigma_c^2}{n}$ se obtiene que:

$$\mu_1 = \frac{\dfrac{\mu_0\sigma_c^2}{n} + \overline{x}\sigma_0^2}{\dfrac{\sigma_c^2}{n} + \sigma_0^2} = \frac{\mu_0\tau^2 + \overline{x}\sigma_0^2}{\tau^2 + \sigma_0^2} = \frac{\tau^2}{\tau^2 + \sigma_0^2}\mu_0 + (1 - \frac{\tau^2}{\tau^2 + \sigma_0^2})\overline{x} \tag{2.26}$$

Como se puede ver se ha llegado a una expresión donde la media de la distribución final se encuentra en una ponderación de la media de la distribución inicial y de la media muestral. Desde el punto de vista bayesiano, los pesos de ecuación 2.26 coinciden con los dados en la definición 2.2.5 del codificador James-Stein a excepción de su interpretación, en la estadística bayesiana μ_0, τ^2 o σ_0^2 y σ_c^2 son conocidos y establecidos a priori, mientras que en la rama de

estudios Empirical Bayes estos se establecen a partir de los datos.

Una vez descrito todo el desarrollo puede observarse que este codificador se ha desarrollado para un problema de regresión donde la variable a predecir Y se distribuye siguiendo una Normal, por tanto no es adecuado utilizar este codificador para problemas de clasificación o problemas donde Y no siga una Normal.

Este codificador no se aplicará al conjunto de datos reales debido a que la variable target, *Objetivo* es de tipo binario. Aunque se puede llegar a utilizar este codificador para este caso, reemplazando la media de Y mediante los log-odds ratio es preferible utilizar otro codificador como es Weight of Evidence, que se definirá más adelante.

2.2.6. Quantile Encoder y Summary Encoder

A lo largo de este trabajo se han descrito varios codificadores los cuales se apoyan en la utilización de la media. Sin embargo, en problemas donde la distribución de los datos presenten grandes colas esto puede llevar, debido a las propiedades de la media, a una mala codificación de la variable categórica y por tanto tener una peor capacidad de predicción.

Por este motivo los autores del artículo [16] proponen un codificador que en lugar de utilizar la media se basa en la utilización de los cuantiles, de ahí el nombre de este método para tratar las variables categóricas. Este encoder resulta ser mejor y más flexible que el Target Encoder (2.2.1) para problemas de regresión con un conjunto de datos con gran dimensionalidad y evaluando la calidad de la predicción mediante el error absoluto medio.

La idea detrás del Quantile Encoder es apoyarse en el desarrollo de Target Encoder pero conseguir una codificación mediante el uso de cuartiles. Una primera idea sería sustituir cada categoría de la variable categórica X_j por el p-cuantil de la siguiente forma:

$$[x_i^j] \longmapsto q_p(y_{[x_i^j]}) \quad \forall i = 1, ..., n_{TR} \tag{2.27}$$

Donde $q_p(y_{[x_i^j]})$ sería el p-cuantil de la variable target y para cada categoría de X_j. Sin embargo mediante esta codificación, al igual que con el codificador Target Encoder, para categorías con muy poca cardinalidad la codificación contendría información muy cercana a y y se tendría el peligro de cometer sobreajuste. Por tanto, para controlarlo se utiliza, al igual que en el codificador m-Estimate 2.2.4, un parámetro de regularización m. De esta forma se define el Quantile Encoder de la siguiente manera:

Definición 2.2.6 (Quantile Encoder)**.** *El Quantile Encoder es una aplicación,*

$$\begin{aligned} \varphi_{QE} : X_j &\longrightarrow \mathbb{R} \\ x_i^j &\longmapsto \varphi_{QE}(x_i^j) = \frac{n_{[x_i^j]} q_p(y_{[x_i^j]}) + q_p(y)m}{n_{[x_i^j]} + m} \quad \forall i = 1, ..., n_{TR} \end{aligned} \tag{2.28}$$

Donde:

- $n_{[x_i^j]}$ *es el número de observaciones de la categoría de* x_i *en* X_j*.*
- $q_p(y_{[x_i^j]})$ *es el p-cuantil de la variable a predecir* y*, para cada categoría de la variable categórica* X_j*.*
- $q_p(y)$ *es el p-cuantil de la variable a predecir* y*.*
- $p \in [0, 1]$ *es el valor del cuantil de la distribución de* y*.*
- $m \geq 0$ *es un hiperparámetro de regularización a establecer.*

Observación 2.2.9. *En el caso particular de usar* $p = 0{,}5$ *equivaldrá a utilizar la mediana de* y*. Además si se utiliza* $m = 0$ *corresponderá a no utilizar regularización alguna.*

Proposición 2.2.2. *El Quantile Encoder con* $p = 0{,}5$ *(Median Encoder) es mejor que otros codificadores que usan la media para problemas de regresión donde se evalúa el error mediante el error absoluto medio.*

Esta proposición se basa en que mientras que para el error cuadrático medio la media es el estimador que lo minimiza, para el error absoluto medio es la mediana. En el artículo mencionado al principio, se compara Quantile Encoder con Target Encoder con varios conjuntos de datos y se realiza un estudio de significancia para comprobar si la proposición 2.2.2 es cierta, pudiendo rechazar la hipótesis nula de que ambos codificadores tienen los mismos resultados, siendo en todos mejor el Quantile Encoder.

Continuando con el siguiente codificador que se menciona en el título de esta sección, se estudia el Summary Encoder. Se trata de en lugar de codificar cada categoría mediante un único cuantil, poder reemplazar la categoría por varios cuantiles y así permitir al modelo poder aprender mejor cómo es la distribución de y para cada una de estas categorías de la variable categórica. Por tanto se define el Summary Encoder como:

Definición 2.2.7 (Summary Encoder)**.** *El Summary Encoder es una aplicación,*

$$\begin{aligned} &\varphi_{SE} : X_j \longrightarrow \mathbb{R}^{\rho} \\ &\qquad x_i^j \longmapsto \varphi_{SE}(x_i^j) = \frac{n_{[x_i^j]} q_p(y_{[x_i^j]}) + q_p(y) m_p}{n_{[x_i^j]} + m_p} \quad \forall i = 1, ..., n_{TR} \quad \forall p \in P \end{aligned} \tag{2.29}$$

Donde:

- $n_{[x_i^j]}$ *es el número de observaciones de la categoría de* x_i *en* X_j*.*
- $q_p(y_{[x_i^j]})$ *es el p-cuantil de la variable a predecir* y*, para cada categoría de la variable categórica* X_j*.*
- $q_p(y)$ *es el p-cuantil de la variable a predecir* y*.*
- P *es el conjunto de cuantiles que se quiere calcular.*
- $p \in [0, 1]$ *es el valor del cuantil de la distribución de* y*.*
- ρ *es el tamaño del conjunto* P*.*
- $m_p \geq 0$ *es un hiperparámetro de regularización a establecer para cada uno de los cuantiles a calcular.*

Observación 2.2.10. *Como se puede ver, al codificar cada categoría por más de un valor, se aumenta la dimensionalidad del problema además, también el número de hiperparámetros a calcular ya que para cada p hay un nuevo* m_p*.*

Una opción es fijar $m = 0$ y no realizar regularización, reduciendo la complejidad de la búsqueda de hiperparámetros pero controlando que no se produzca sobreajuste ya que cuanto mayor sea ρ mayor riesgo habrá de cometerlo.

Observación 2.2.11. *Estos dos codificadores, debido a que se basan en el cálculo de cuantiles de la variable objetivo, son una opción a considerar cuando se trata de un problema de regresión y no de clasificación.*

De esta forma se han descrito otros dos métodos para codificar variables categóricas que son más robustos que los que se basan en medias para cuando hay muchos outliers en el conjunto de datos o cuando existen colas pronunciadas en la distribución de la variable y.

2.2.7. Weight of Evidence Encoder

Este codificador para tratar variables categóricas es empleado también como método para discretizar variables, es muy utilizado en ambientes como el Credit Scoring donde por ley, las entidades bancarias tienen que justificar si deciden no conceder un crédito a un cliente. Esto lleva a ver que el uso de este codificador estará reservado para problemas de tipo supervisado en el que la variable y es binaria, es decir, un problema de clasificación binaria.

Centrándose en el uso de este codificador como método para tratar variables categóricas, se define como variable WOE la siguiente transformación de la variable X_j:

$$[x_i^j] \longmapsto -log\left(\frac{\frac{P(Y=0)}{P(Y=0|X_j=[x_i^j])}}{\frac{P(Y=1)}{P(Y=1|X_j=[x_i^j])}}\right) \tag{2.30}$$

Sin embargo, estas probabilidades lo habitual es no conocerlas y por tanto hay que estimarlas apoyándose en el conjunto de entrenamiento. En la codificación 2.30 se requiere el cálculo de cuatro probabilidades pero desarrollando el argumento del logaritmo, es posible llegar a una expresión en términos únicamente de dos probabilidades y por tanto dado a que como se ha dicho anteriormente hay que estimarlas, será más conveniente:

$$\begin{aligned}
-log\left(\frac{\frac{P(Y=0)}{P(Y=0|X_j=[x_i^j])}}{\frac{P(Y=1)}{P(Y=1|X_j=[x_i^j])}}\right) &= -log\left(\frac{\frac{P(Y=0)P(X_j=[x_i^j])}{P(Y=0\cap X_j=[x_i^j])}}{\frac{P(Y=1)P(X_j=[x_i^j])}{P(Y=1\cap X_j=[x_i^j])}}\right) \\
&= -log\left(\frac{\frac{P(Y=0)P(X_j=[x_i^j])}{P(X_j=[x_i^j]|Y=0)P(Y=0)}}{\frac{P(Y=1)P(X_j=[x_i^j])}{P(X_j=[x_i^j]|Y=1)P(Y=1)}}\right) \\
&= -log\left(\frac{P(X_j=[x_i^j]|Y=1}{P(X_j=[x_i^j]|Y=0}\right) \\
&= log\left(\frac{P(X_j=[x_i^j]|Y=0}{P(X_j=[x_i^j]|Y=1}\right)
\end{aligned} \tag{2.31}$$

Esta última expresión 2.31 en término de dos probabilidades condicionadas, es la que finalmente se utiliza en el Weight of Evidence (WOE) Encoder.

Observación 2.2.12. *Nótese que existe la posibilidad de que el denominador del argumento del logaritmo sea 0.*

Al realizar la estimación de $P(X_j = [x_i^j]|Y = 1)$ puede darse la situación de que valga cero y no se pueda realizar la codificación, es por ello que para evitar este problema, se añade un término de regulación. Por tanto, la definición del WOE Encoder queda de la siguiente forma:

Definición 2.2.8 (Weight of Evidence Encoder)**.** *El Weight of Evidence Encoder es una aplicación,*

$$\begin{aligned} &\varphi_{WOE} : X_j \longrightarrow \mathbb{R} \\ &x_i^j \longmapsto \varphi_{WOE}(x_i^j) = log\left(\frac{\frac{n_{[x_i^j],Y=0}+r}{n_{Y=0}}}{\frac{n_{[x_i^j],Y=1}+r}{n_{Y=1}}}\right) \quad \forall i = 1, ..., n_{TR} \end{aligned} \tag{2.32}$$

Donde:

- $n_{[x_i^j],Y=1}$ *es el número de observaciones de la categoría de* x_i *en* X_j *con* $Y = 1$*.*
- $n_{[x_i^j],Y=0}$ *es el número de observaciones de la categoría de* x_i *en* X_j *con* $Y = 0$*.*
- $n_{Y=1}$ *es el número de observaciones con* $Y = 1$*.*
- $n_{Y=0}$ *es el número de observaciones con* $Y = 0$*.*
- r *es el parámetro de regularización, habitualmente es* $0,5$ *o* 1*.*

Debido al tipo de problema al que está ligado este codificador y a las situaciones en las que se suele emplear, es común utilizarlo en el modelo de Regresión Logística. Además de todo esto, existe una razón de peso por la que se utilizan variables WOE en este tipo de modelos. El tratamiento de la variable categórica mediante el codificador Weight of Evidence genera una variable WOE, la cual está relacionada con los ODDS de la Regresión Logística.

Sea $\pi(x) = P(Y = 1|X = x)$ la probabilidad de que la variable target Y valga 1 condicionada a que X valga x. Si se toma como $logit(\pi(x)) = log\left(\frac{\pi(x)}{1-\pi(x)}\right)$, esta función tomará valores dentro de los reales por lo que se puede plantear una regresión lineal de la forma:

$$logit(\pi(x)) = \alpha + \beta X + \epsilon'$$

Desarrollando esta ecuación:

$$\begin{aligned} log\left(\frac{\pi(x)}{1-\pi(x)}\right) = \alpha + \beta X + \epsilon' &\iff \\ \frac{\pi(x)}{1-\pi(x)} = e^{\alpha} + \beta X + \epsilon' &\iff \\ \pi(x) = \frac{1}{1+e^{-(\alpha+\beta X+\epsilon)}} & \end{aligned} \tag{2.33}$$

De esta forma se llega a lo que se conoce como modelo de Regresión Logística con el uso de la función logit como función linkaje. Si se quiere realizar una interpretación de los parámetros estimados, no es posible hacerlo de forma directa ya que la ecuación 2.33 no es lineal. Es aquí donde intervienen el ODDS y los ODDS ratios.

Supóngase que se tiene una variable independiente X y la variable binaria Y. Sea $Z = \alpha + \beta X$ y se plantea la regresión logística que estima la probabilidad de $Y = 1$ dado X:

$$\begin{aligned} P(Y=1|X=x) &= \frac{1}{1+e^{-z}} \iff \\ P(Y=1|X=x)(1+e^{-z}) &= 1 \iff \\ P(Y=1|X=x)e^{-z} &= 1 - P(Y=1|X=x) \iff \\ e^{-z} &= \frac{1-P(Y=1|X=x)}{P(Y=1|X=x)} \iff \\ e^{z} &= \frac{P(Y=1|X=x)}{1-P(Y=1|X=x)} = e^{\alpha}e^{\beta x} \end{aligned} \tag{2.34}$$

La medida 2.34 es lo que se conoce como ODDS e indica para un valor x cuántas veces es más probable que $Y = 1$ a que $Y = 0$. En cambio, los ODDS ratios (OR) o razón de ODDS:

$$OR = \frac{ODDS(x+1)}{ODDS(x)} = \frac{e^{\alpha}e^{\beta(x+1)}}{e^{\alpha}e^{\beta x}} = e^{\beta} \tag{2.35}$$

Este factor 2.35 indica cómo se incrementa el ratio de comparar la probabilidad de $Y = 1$ frente a $Y = 0$ cuando se incrementa en una unidad el valor de la variable explicativa. Si OR $> 1 \implies \beta > 0$ indica que la variable X aumenta la probabilidad de $Y = 1$, en cambio, si OR $< 1 \implies \beta < 0$ indica que la variable X disminuye la probabilidad de $Y = 1$.

Proposición 2.2.3. *La variable WOE es proporcional al ODDS de la Regresión Logística.*

Demostración. Utilizando la expresión 2.30 para la codificación de la variable categórica:

$$\begin{aligned} -log\left(\frac{\frac{P(Y=0)}{P(Y=0|X_j=[x_i^j])}}{\frac{P(Y=1)}{P(Y=1|X_j=[x_i^j])}}\right) &= -log\left(\frac{P(Y=1|X_j=[x_i^j])}{P(Y=0|X_j=[x_i^j])}\right) + log\left(\frac{P(Y=1)}{P(Y=0)}\right) \\ &\propto log\left(\frac{P(Y=1|X_j=[x_i^j])}{P(Y=0|X_j=[x_i^j])}\right) = logit(\pi(x)) \\ &= log(ODDS_{[x_i^j]}) \end{aligned}$$

□

De esta forma se consigue una codificación mediante una escala que es natural para el modelo de Regresión Logística, por lo que si se va a emplear para resolver el problema de clasificación binaria, puede ser útil este tipo de codificación para las variables categóricas.

Aplicando el WOE Encoder al conjunto del estudios, al ser un problema de clasificación binaria, encaja perfectamente. Tomando como ejemplo la variable *Comunidad Autónoma Familiar*:

	Comunidad Autónoma Familiar	**WOE Encoder**
1	MADRID	−0,018315
2	LA RIOJA	−0,263829
3	MADRID	−0,018315
4	MADRID	−0,018315
5	MADRID	−0,018315
6	MADRID	−0,018315
7	ANDALUCIA	0,141601
⋮	⋮	⋮

Tabla 10: Ejemplo de variable codificada mediante WOE Encoder.

Como en el resto de los codificadores supervisados, el tiempo en codificar las variables categóricas es bajo, en concreto 2,390438 segundos. Además también mantiene la memoria utilizada con respecto al conjunto de datos original.

A modo de resumen se crea una tabla donde se reflejan aspectos de los codificadores sobre el conjunto de datos real. Estos utilizan $21,1$ MB de memoria y tienen un total de 30 variables, de las cuales 18 son categóricas.

Encoders	**T. Ejecución (s)**	**Nº columnas**	**Memoria**	**Hiperpar.**	**Imagen**
Ordinal	$1,181$	18	$21,1$ MB	0	$\mathbb{N}^{18}$
One Hot	$> 30,152$	8048	$> 2,4$ GB	0	$\mathbb{Z}_2^{8048}$
Dummy	$> 30,152$	8047	$> 2,4$ GB	0	$\mathbb{Z}_2^{8047}$
Binary	$3,195$	119	$83,8$ MB	0	$\mathbb{Z}_2^{119}$
Hashing	$64,482$	$c_0 = 8$	$15,5$ MB	c_0	$\mathbb{N}^{c_0}$
Frequency	$2,600$	18	$21,1$ MB	0	$(0,1)^{18}$
Target	$2,699$	18	$21,1$ MB	$h = 20$, $f = 10$	$[0,1]^{18}$
LOO	$1,703$	18	$21,1$ MB	0	$[0,1]^{18}$
CatBoost	$1,574$	18	$21,1$ MB	$a = 1$	$[0,1]^{18}$
m-estimate	$2,353$	18	$21,1$ MB	$m = 1$	$[0,1]^{18}$
WOE	$2,290$	18	$21,1$ MB	0	$\mathbb{R}^{18}$

Tabla 11: Tabla resumen de los codificadores sobre el conjunto de datos.

De esta forma se pueden apreciar las principales diferencias entre los distintos métodos para tratar las variables categóricas aplicados al conjunto de datos.

3. Estudio en un Caso Real

3.1. Planteamiento del Problema

Una vez vista la teoría en la que se basan estos codificadores de variables categóricas, se seleccionarán algunos para estudiar sus implicaciones a nivel práctico en los modelos de Machine Learning. Se utilizará como conjunto de datos los mencionados al inicio del trabajo, en la sección 2. Esto implica que algunos codificadores que no están diseñados para problemas de clasificación, en concreto, binaria, quedan descartados.

Para el estudio, se incluirán los siguientes codificadores por las siguientes razones:

- (2.1.3) **Binary Encoder**: Al no poder utilizar el One Hot Encoder por el gran aumento de la dimensionalidad y en consecuencia por la cantidad de memoria que consume, es el método que más se le asemeja.
- (2.1.4) **Hashing Encoder**: Es interesante comprobar qué tal se comportan los modelos con un método que reduce tanto la dimensionalidad.
- (2.1.5) **Frequency Encoder**: Es uno de los métodos más directos y que al codificar las variables en el intervalo $[0, 1]$ puede funcionar bien para algunos modelos como las Redes Neuronales.
- (2.2.1) **Target Encoder**: Es el codificador supervisado más extendido.
- (2.2.3) **CatBoost Encoder**: Como se vio es un codificador interesante que conviene estudiar junto a su respectiva variante del Gradient Boosting.
- (2.2.7) **Weight of Evidence Encoder**: Es un codificador famoso que es popular por su relación con la Regresión Logística.

Para cada uno de estos codificadores se emplearán modelos basados en diferentes conceptos. Por un lado se estudiará cómo afectan los codificadores mencionados anteriormente en modelos basados en árboles de decisión, para ello se selecciona el Gradient Boosting, así de esta forma también se podrá comparar con CatBoost, mientras que por parte de las Redes Neuronales se escogerá el Perceptrón Multicapa (MLP). También se estudiará la Maquina de Soporte Vectorial (SVM) y por último la Regresión Logística.

Para realizar el estudio se emplearán las siguientes librerías de Python: Scikit-learn [18], CatBoost [6] y Category Encoder [14] en el entorno de Google Colab. Para poder comparar los modelos, se dejarán por defecto los hiperparámetros de estos. A continuación se seleccionará el mejor modelo y se ajustarán.

Lo primero que destaca en este problema es que la variable *Objetivo* se encuentra desbalanceada, esto puede ser un aspecto interesante para ver si alguna codificación ayuda al modelo a aprender sobre el conjunto de entrenamiento. En un inicio no se balanceará y sobre el modelo final se estudiará si conviene balancear las clases de la variable *Objetivo*.

A continuación se hace un estudio de los valores missing y se observa que en las dos variables categóricas ordinales *O1* y *O2* falta una gran cantidad de valores, en ambas roza casi un cuarto de los datos. Por ello, al no conocer el origen de los datos se opta por considerar estos valores como una categoría más dentro de estas variables ordinales y tratarlas como si fuesen nominales. También la variable *Provincia Nacimiento* cuenta con un valor faltante que al ser solo uno, se elimina esta observación del estudio.

Por último se dividen los datos del estudio en conjunto de entrenamiento y conjunto de test. Con respecto a los posibles outliers, al no conocer el significado de las variables se decide hacer una división por rango, llevando los valores de las variables numéricas al intervalo $[0, 1]$ y de esta forma mitigar la influencia de los outliers.

Una vez establecido el modo de operar falta determinar la métrica a utilizar para comparar modelos. Debido a que no se conoce exactamente la finalidad del modelo, se calcularán las siguientes tres métricas:

- Exactitud: Se trata de la proporción total de clases asignadas correctamente:

$$Accuracy = \frac{VP + VN}{VP + VN + FN + FV}$$

  Es importante no confundirla con la precisión.

- Log-loss o Entropía Cruzada:

$$\mathcal{L}(Y, P) = -\frac{\sum_{i=1}^{n_T} y_i log(p_i) + (1 - y_i) log(1 - p_i)}{n_T}$$

  Donde $y_i \in 0, 1$ es la clase de la observación i-ésima del conjunto de test y p_i es la probabilidad estimada $P(y_i = 1)$.

- Área bajo la curva ROC (AUC).

3.2. Resultados de las Comparaciones

Una vez entrenado los modelos con el conjunto de entrenamiento, se emplea el conjunto de test para evaluar el rendimiento de los codificadores de variables categóricas en cada uno de los modelos.

3.2.1. Gradient Boosting

Una vez entrenado y evaluado el modelo Gradient Boosting se obtienen los siguientes valores:

CODIFICADOR	TIEMPO EJECUCIÓN (s)	MÉTRICA	VALOR
Binary Encoder	42,8738	Accuracy	0,9198
		Log-loss	0,2226
		AUC	0,7677
Hashing Encoder	13,6363	Accuracy	0,9165
		Log-loss	0,2294
		AUC	0,7605
Frequency Encoder	17,8936	Accuracy	0,9188
		Log-loss	0,2227
		AUC	0,7653
Target Encoder	17,1372	Accuracy	0,9194
		Log-loss	0,2189
		AUC	0,7676
CatBoost Encoder	61,3167	Accuracy	0,9195
		Log-loss	0,2180
		AUC	0,7682
WOE Encoder	16,9937	Accuracy	0,9186
		Log-loss	0,2201
		AUC	0,7654

Tabla 12: Resultados Gradient Boosting.

Se puede observar que no hay mucha diferencia entre codificadores ni en términos de Accuracy ni en AUC.

Sin embargo, fijándose en el Log-loss destacan los codificadores Target Encoder y CatBoost Encoder, lo que puede hacer pensar que sea conveniente utilizar el modelo CatBoost. También cabe destacar como cabía esperar, que el método Binary Encoder provocaría un mayor tiempo de ejecución, sin embargo, es sorprendente el tiempo que necesita el Gradient Boosting con el codificador CatBoost.

Seleccionando el Gradient Boosting con el codificador Target Encoder, queda la siguiente matriz de confusión:

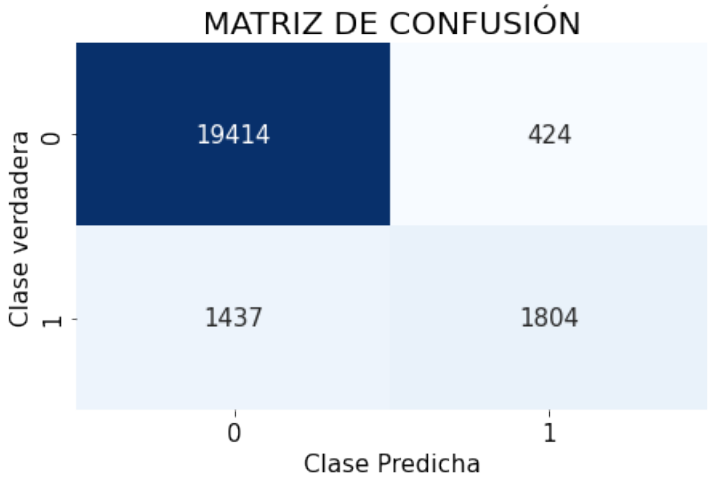

Figura 1: Matriz de confusión del modelo Gradient Boosting con el codificador de variables categóricas Target Encoder.

3.2.2. Perceptrón Multicapa (MLP)

Una vez entrenada y evaluada la red neuronal MLP se obtienen los siguiente valores:

CODIFICADOR	TIEMPO EJECUCIÓN (s)	MÉTRICA	VALOR
Binary Encoder	193,0192	Accuracy	0,9076
		Log-loss	0,2505
		AUC	0,7887
Hashing Encoder	139,2521	Accuracy	0,9133
		Log-loss	0,2380
		AUC	0,7585
Frequency Encoder	148,2114	Accuracy	0,9102
		Log-loss	0,2480
		AUC	0,7368
Target Encoder	163,2363	Accuracy	0,9126
		Log-loss	0,2349
		AUC	0,7497
CatBoost Encoder	130,5772	Accuracy	0,9116
		Log-loss	0,2342
		AUC	0,7483
WOE Encoder	155,8009	Accuracy	0,9136
		Log-loss	0,2296
		AUC	0,7583

Tabla 13: Resultados MLP.

En este modelo no hay un claro vencedor ya que según la métrica que se seleccione convendría un codificador u otro. Sin embargo, sí parece razonable a pesar del tiempo de ejecución preferir el codificador Binary Encoder debido a que hay una cierta diferencia con respecto a los demás en cuanto al área bajo la curva ROC.

Seleccionando el método Binary Encoder, se obtiene la siguiente matriz de confusión:

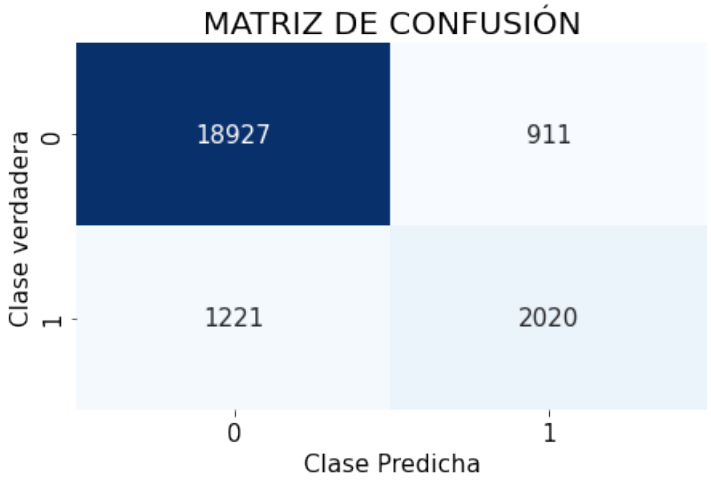

Figura 2: Matriz de confusión del modelo MLP con el codificador de variables categóricas Binary Encoder.

3.2.3. Máquina de Soporte Vectorial (SVM)

Una vez entrenada y evaluada la Máquina de Soporte Vectorial se obtienen los siguientes valores:

CODIFICADOR	TIEMPO EJECUCIÓN (s)	MÉTRICA	VALOR
Binary Encoder	438,0805	Accuracy	0,9098
		AUC	0,7361
Hashing Encoder	107,3015	Accuracy	0,9019
		AUC	0,7025
Frequency Encoder	145,8275	Accuracy	0,9067
		AUC	0,7203
Target Encoder	146,7960	Accuracy	0,9127
		AUC	0,7419
CatBoost Encoder	160,8309	Accuracy	0,9120
		AUC	0,7410
WOE Encoder	156,9855	Accuracy	0,9081
		AUC	0,7301

Tabla 14: Resultados SVM.

Lo primero que hay que mencionar es que como el modelo SVM no se basa en probabilidades, no se puede utilizar la métrica Log-loss. Mediante este método se obtienen peores resultados que con el Gradiente Boosting y con la MLP. Otro aspecto muy destacable es el tiempo de ejecución que resulta de utilizar para este modelo el Binary Encoder siendo este muy elevado y obteniendo peores resultados que con Target Encoder.

Seleccionando el codificador Target Encoder resulta la siguiente matriz de confusión:

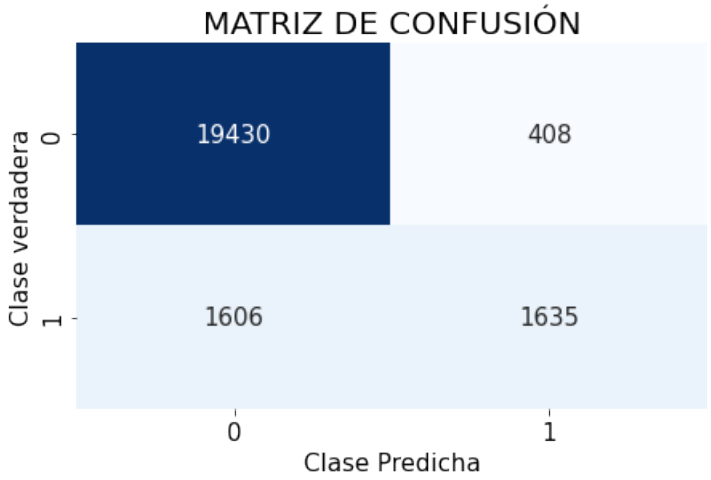

Figura 3: Matriz de confusión del modelo SVM con el codificador de variables categóricas Target Encoder.

3.2.4. Regresión Logística

Una vez entrenado y evaluado el modelo de Regresión Logística se obtienen los siguientes valores:

CODIFICADOR	TIEMPO EJECUCIÓN (s)	MÉTRICA	VALOR
Binary Encoder	3,1912	Accuracy	0,9089
		Log-loss	0,2423
		AUC	0,7462
Hashing Encoder	1,4077	Accuracy	0,9065
		Log-loss	0,2527
		AUC	0,7402
Frequency Encoder	1,7865	Accuracy	0,9050
		Log-loss	0,2576
		AUC	0,7341
Target Encoder	1,7111	Accuracy	0,9042
		Log-loss	0,2460
		AUC	0,7344
CatBoost Encoder	1,9212	Accuracy	0,9055
		Log-loss	0,2450
		AUC	0,7369
WOE Encoder	1,7219	Accuracy	0,9057
		Log-loss	0,2465
		AUC	0,7359

Tabla 15: Resultados Regresión Logística.

Con el modelo de Regresión Logística sucede de forma similar que con el SVM, se obtienen peores resultados que con el Gradient Boosting y que con la MLP por tanto este modelo también sería descartable salvo que se quisiera un modelo que fuese interpretable, en ese caso habría que utilizarlo y sería recomendable usar Binary Encoder. Aunque con este codificador es más lento, la Regresión Logística funciona mucho más rápido que el resto de modelos.

Usando la Regresión Logística como clasificador y utilizando Binary Encoder como método para tratar las variables categóricas, se obtiene la siguiente matriz de confusión:

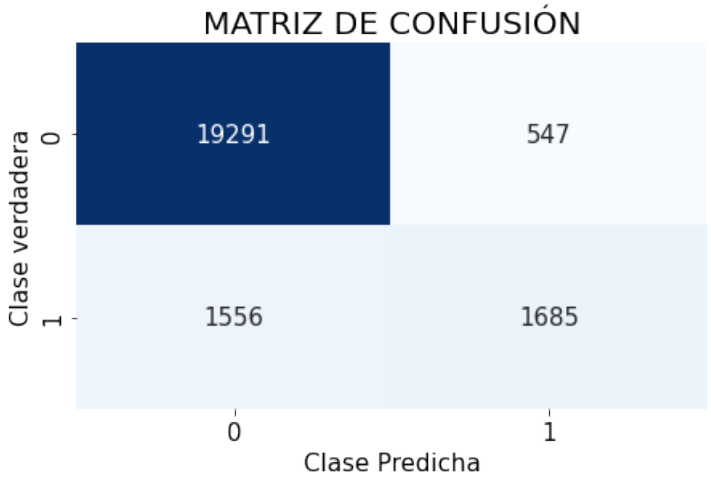

Figura 4: Matriz de confusión del modelo Regresión Logística con el codificador de variables categóricas Binary Encoder.

3.3. CatBoost

Basándose en los resultados obtenidos en el Gradient Boosting con el codificador CatBoost, se va a utilizar la variante de este modelo [6] como clasificador que, como su propio nombre indica, emplea ese codificador de variables categóricas.

Se utiliza este modelo tanto balanceando las clases como manteniendo los datos originales y se comprueba que dependiendo de la métrica a utilizar es mejor utilizar los datos equilibrados o los originales. Utilizando CatBoost con los datos balanceados, con una profundidad en los árboles de 6 y una tasa de aprendizaje de 0,0662 se obtienen los siguientes resultados:

- Accuracy: 0,8659
- Log-loss: 0,3318
- AUC: 0,8525

Con la siguiente matriz de confusión:

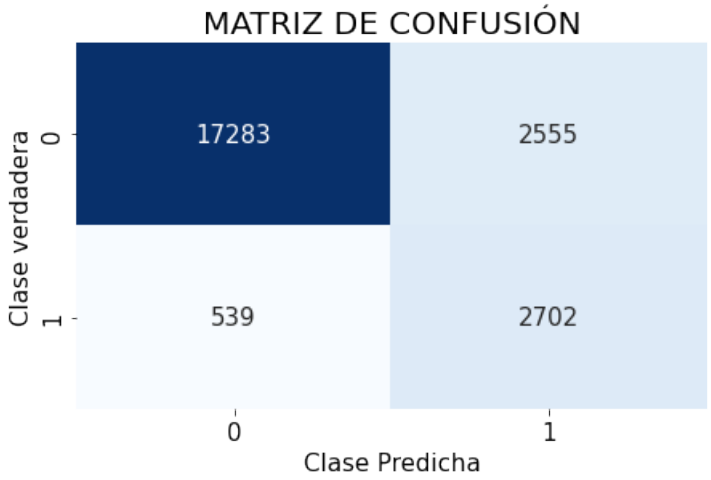

Figura 5: Matriz de confusión del modelo CatBoost con los datos balanceados.

Mientras que con los datos balanceados empleando una profundidad en los árboles de 6 y una tasa de aprendizaje de 0,0903 se obtienen los siguientes resultados:

- Accuracy: 0,9246
- Log-loss: 0,2034
- AUC: 0,7900

Con la siguiente matriz de confusión:

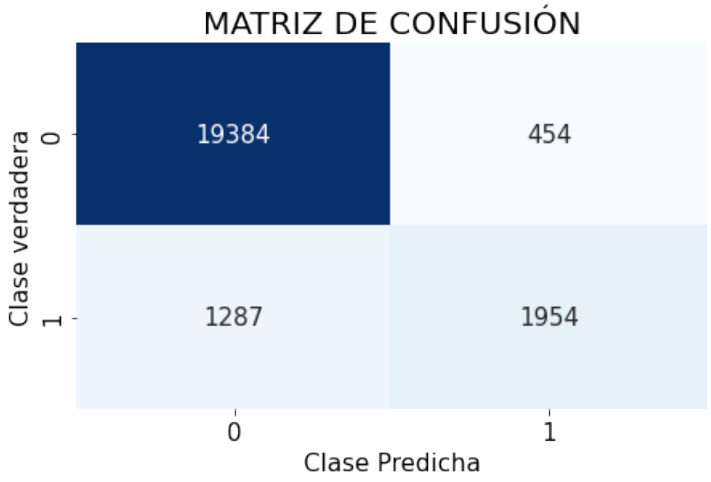

Figura 6: Matriz de confusión del modelo CatBoost con los datos sin balancear.

Si interesa poder clasificar más observaciones con clase 1 será recomendable utilizar el primer modelo, mientras si lo que conviene es tener una buena relación entre las probabilidades de ambas clases la métrica a utilizar será el Log-loss y por tanto se deberá utilizar el segundo modelo.

3.4. Perceptrón Multicapa

Apoyándose en los resultados de la comparativa de modelos y codificadores de variables categóricas, la combinación que mejor AUC obtuvo fue la Red Neuronal Multicapa utilizando el Binary Encoder como codificador de variables categóricas.

Equilibrando las clases mediante bajomuestreo y utilizando una MLP con función de activación sigmoide, una capa oculta con 50 neuronas, con ratio de aprendizaje adaptativo, 500 iteraciones y utilizando como método de optimización el descenso de gradiente estocástico se obtienen los siguientes resultados:

- Accuracy: 0,8331
- Log-loss: 0,4148
- AUC: 0,8103

Con la siguiente matriz de confusión:

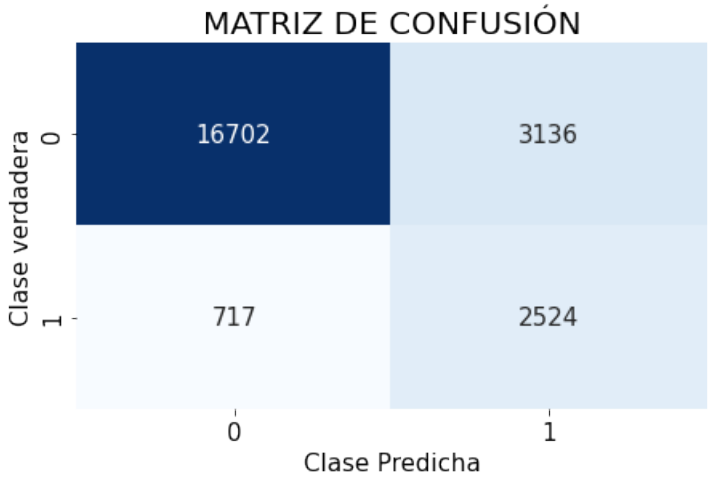

Figura 7: Matriz de confusión del modelo MLP con parámetros mencionados anteriormente y los datos bajomuestreados.

Mejoran los resultados con respecto a la MLP utilizada en la comparativa de modelos y codificadores. A pesar de ello, el modelo CatBoost obtiene mejores resultados en todas las métricas por lo que finalmente se decide que ese modelo es el mejor para abordar este problema.

Una vez obtenido todos los resultados del estudio combinando modelos y codificadores, cabe preguntarse si hay algún motivo por el cual para cada modelo, las métricas difieren muy poco unas de otras variando el método para tratar las variables categóricas.

Haciendo un estudio de las variables más importantes que intervienen en el conjunto de árboles del ensamblado para cada codificador, se puede observar que la influencia de las variables categóricas en la variable *Objetivo* es poca, y la gran mayoría la tienen las variables numéricas, en especial las variables *N09* y *N07*.

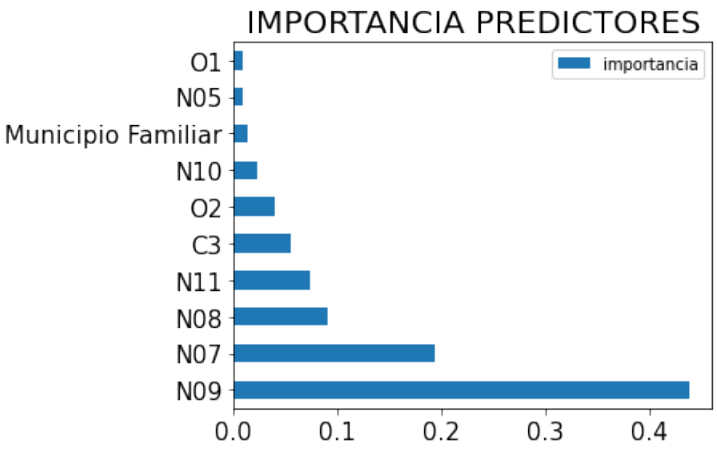

Figura 8: Gráfico de las 10 variables con más importancia en el modelo de Gradient Boosting con el codificador Target Encoder.

Por tanto, esta puede ser la razón del porqué hay poca diferencia en las métricas para cada codificador y por ende, no se aprecie bien qué métodos convendría utilizar para tratar las variables categóricas en estos datos.

Una vez llegado a este punto, después de recabar información en distintos artículos, varios de ellos recientes, y realizar el código para las distintas pruebas sobre el conjunto de datos, por motivos de extensión del trabajo no es posible seguir profundizando en el estudio, sin embargo, a vista de estos resultados, una posible línea futura de análisis sería tratar de clasificar la variable *Objetivo* sin las variables numéricas, pese a que los resultados probablemente serían peores, puede que sí hubiese más diferencia entre los distintos codificadores. Otra línea de estudio sería tratar las variables categóricas de manera independiente para, de esta forma, utilizar distintos métodos para cada una de las variables y ver si mejoran los resultados ya obtenidos.

4. Conclusiones

Una vez estudiado tanto a nivel teórico como práctico queda de manifiesto que no hay una forma óptima de codificar las variables categóricas para cualquier escenario. Se trata de atender a las propiedades de cada codificador y seleccionar aquellas que se adecúen al tipo de problema que se tiene, al modelo a utilizar y a los tipos de datos que se disponen. De esta forma se podrán aprovechar las fortalezas y mitigar las desventajas de cada encoder.

A lo largo de cada codificador se han presentado y descrito observaciones que pueden guiar a realizar un buen uso de estos. De esta forma, una manera de proceder es la siguiente, lo primero será decidir si la variable categórica es nominal u ordinal, en caso de tener orden una opción a considerar es utilizar el Ordinal Encoder, si no es así o este orden no está muy claro este método queda descartado y habrá que fijarse en el número de categorías que tiene dicha variable. Si tiene un número razonable de ellas y por tanto, no causa problemas de memoria por el aumento de la dimensionalidad entonces utilizar One Hot Encoder, en caso contrario considerar utilizar Binary Encoder o Hashing Encoder. En todos estos supuestos Frequency Encoder siempre es otra opción que se tiene.

Las decisiones entre los codificadores mencionados en el párrafo anterior han sido aconsejadas en función de cómo era la variable categórica. Teniendo en cuenta ahora además el tipo de problema que se tiene, si el problema es de tipo supervisado y por tanto se tiene una variable objetivo, se abre un nuevo abanico de posibilidades, estos son los codificadores supervisados. En tal caso siempre se puede considerar los codificadores Target, LOO, CatBoost y m-Estimate Encoder. Particularizando en el tipo de problema supervisado, si se trata de un problema de clasificación binaria habrá que considerar además el codificador Weight of Evidence, si por el contrario se trata de un problema de regresión, entonces las nuevas opciones serán James-Stein, Quantile y Summary Encoder.

Por último, un aspecto a destacar tras del estudio teórico y práctico es que la variante del algoritmo Gradient Boosting, propuesta por los autores del CatBoost Encoder, debe ser una opción a considerar por el buen uso que realiza de las variables categóricas y los buenos resultados del modelo.

Referencias

[1] Subin An. *"11 Categorical Encoders and Benchmark"*. (2019). URL: https://www.kaggle.com/code/subinium/11-categorical-encoders-and-benchmark/notebook.

[2] Deepanshu Bhalla. *"Weight of Evidence (WOE) and Information Value (IV) explained"*. (2015). URL: https://www.listendata.com/2015/03/weight-of-evidence-woe-and-information.html.

[3] Huy Bui. *"How to Encode Categorical Data"*. (2020). URL: https://towardsdatascience.com/how-to-encode-categorical-data-d44dde313131#bbad.

[4] Bojan Cestnik. "Estimating Probabilities: A Crucial Task In Machine Learning". En: *In Proceedings of ECAI 90, Stockholm, August* (1990).

[5] Bojan Cestnik e Ivan Bratko. "On estimating probabilities in tree pruning". En: *European Working Session on Learning*. Springer. (1991), págs. 138-150. DOI: 10.1007/BFb0017010.

[6] Anna Veronika Dorogush, Vasily Ershov y Andrey Gulin. "CatBoost: gradient boosting with categorical features support". En: *arXiv preprint arXiv:1810.11363* (2018). DOI: 10.48550/arXiv.1810.11363.

[7] Bradley Efron y Carl Morris. "Stein's Paradox in Statistics". En: *Scientific American* 236.5 (1977), págs. 119-127. ISSN: 00368733, 19467087. URL: http://www.jstor.org/stable/24954030.

[8] Python Software Foundation. *"Hashlib — Secure hashes and message digests"*. https://docs.python.org/3/library/hashlib.html.

[9] M.A. Gómez Villegas. *"Inferencia estadística"*. Editorial Díaz de Santos, S.A., 2005. ISBN: 9788479781224. URL: https://books.google.es/books?id=YOuODwAAQBAJ.

[10] John T Hancock y Taghi M Khoshgoftaar. "Survey on categorical data for neural networks". En: *Journal of Big Data* 7.1 (2020), págs. 1-41. DOI: 10.1186/s40537-020-00305-w.

[11] William James y Charles Stein. "Estimation with quadratic loss". En: *Breakthroughs in statistics*. Springer, (1992), págs. 443-460. DOI: 10.1007/978-1-4612-0919-5_30.

[12] Manu Joseph. *"The Gradient Boosters V: CatBoost"*. (2020). URL: https://deep-and-shallow.com/2020/02/29/the-gradient-boosters-v-catboost/.

[13] M. Kuhn y K. Johnson. *"Feature Engineering and Selection: A Practical Approach for Predictive Models"*. Chapman & Hall/CRC Data Science Series. CRC Press, (2019). ISBN: 9781351609470. URL: https://books.google.es/books?id=xy73DwAAQBAJ.

[14] Will McGinnis. *"Category Encoders"*. https://contrib.scikit-learn.org/category_encoders/index.html. (2022).

[15] Daniele Micci-Barreca. "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems". En: *ACM SIGKDD Explorations Newsletter* 3.1 (2001), págs. 27-32. DOI: 10.1145/507533.507538.

[16] Carlos Mougan et al. "Quantile encoder: Tackling high cardinality categorical features in regression problems". En: *International Conference on Modeling Decisions for Artificial Intelligence*. Springer. (2021), págs. 168-180.

[17] Möbius. *"An Overview of Categorical Encoding Methods"*. (2020). URL: https://www.kaggle.com/code/arashnic/an-overview-of-categorical-encoding-methods/notebook.

[18] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.

[19] Liudmila Prokhorenkova et al. "CatBoost: unbiased boosting with categorical features". En: *Advances in neural information processing systems* 31 (2018). URL: https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html.

[20] Adrián Rocha Íñigo. *"Codificación de variables categóricas en aprendizaje automático"*. Universidad de Sevilla, (2020). URL: https://idus.us.es/handle/11441/108887.

[21] Chris Said. *"Empirical Bayes for multiple sample sizes"*. (2017). URL: https://chris-said.io/2017/05/03/empirical-bayes-for-multiple-sample-sizes/.

[22] Cedric Seger. *"An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing"*. (2018). URL: https://www.diva-portal.org/smash/record.jsf?dswid=-2643&pid=diva2%3A1259073.

[23] Charles Stein. "Inadmissibility of the usual estimator for the mean of a multivariate normal distribution". En: *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability: Contributions to the Theory of Statistics*. Vol. 1. University of California Press. (1956), pág. 197.

[24] Paul Westenthanner. *"Unique levels, smoothing, and QuantileEncoder"*. (2022). URL: https://github.com/scikit-learn-contrib/category_encoders/issues/327.