UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Posgrado en Ciencia e Ingeniería de la Computación

Metodología para la representación de atributos categóricos en códigos que preserven patrones y su aplicación en técnicas de aprendizaje automático.

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

DOCTOR EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

P R E S E N T A:

ERIC VALDEZ VALENZUELA

TUTOR:

DR. ÁNGEL FERNANDO KURI MORALES (IIMAS, UNAM)

CO-TUTORA:

DRA. HELENA MONTSERRAT GÓMEZ ADORNO (IIMAS, UNAM)

C.U., Ciudad de México Abril, 2025

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y

HONESTIDAD ACADÉMICA Y PROFESIONAL

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a ala institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado Metodología para la representación de atributos categóricos en códigos que preserven patrones y su aplicación en técnicas de aprendizaje automático, que presenté para obtener el grado de Doctor en Ciencia e Ingeniería de la Computación, es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Programa de Posgrado, citando las fuentes, ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad e los actos de carácter académico administrativo del proceso de titulación/graduación

Atentamente

Eric Valdez Valenzuela

306608610

Jurado asignado:

Presidente:	Dr. Boris Escalante Ramírez
Vocal:	Dr. Ángel Fernando Kuri Morales
Secretario:	Dr. Gibrán Fuentes Pineda
Suplente:	Dra. Helena Montserrat Gómez Adorno
Suplente:	Dra. Wendy Elizabeth Aguilar Martínez

Director de tesis:

_____

Dr. Ángel Fernando Kuri Morales

Agradecimientos

Agradezco el financiamiento de la beca de doctorado otorgada por el CONAHCYT (anteriormente CONACYT). Asimismo, agradezco el apoyo financiero del proyecto DGAPA UNAM-PAPIIT número IN104424 y del proyecto CONAHCYT número CF-2023-G-64. Finalmente, expreso mi gratitud al Posgrado en Ciencia e Ingeniería de la Computación por la oportunidad de cursar el programa de doctorado en el campo de Inteligencia Artificial.

Ciudad Universitaria, IIMAS, 2025

Índice general

Índice de figuras

Índice de tablas

Metodología para la representación de atributos categóricos en códigos que preserven patrones y su aplicación en técnicas de aprendizaje automático.

Eric Valdez Valenzuela
Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México

Resumen

El objetivo de este trabajo de investigación se enfoca en cómo tratar los atributos categóricos presentes en las bases de datos para su aplicación en técnicas de aprendizaje automático (ML por sus siglas en inglés). Los atributos categóricos son aquellos que contienen un número finito de valores o grupos distintos. Por otro lado, la mayoría de los métodos de ML, en la práctica, trabajan únicamente con atributos numéricos, por lo que es necesario aplicar un procesamiento a los atributos categóricos para representarlos en un espacio métrico. A pesar de que existen múltiples técnicas para convertir atributos categóricos a un valor numérico, si no se aplican de manera correcta, es posible que no se conserven los patrones existentes en los atributos categóricos. Esta pérdida de información afecta el desempeño de los modelos de ML, por lo que es de primordial importancia conservar los posibles patrones embebidos en las variables categóricas.

La información contenida en los atributos categóricos se puede representar mediante Códigos que Preserven los Patrones (PPC). Existen técnicas de codificación que encuentran estos PPCs, una de ellas es CESAMO: *Categorical Encoding By Statistical Applied Modeling*. En esta investigación se propone estudiar y ampliar el conocimiento de esta técnica. Uno de los principales objetivos fue determinar cómo las distintas técnicas de codificación afectan a los algoritmos de ML. Para ello se evaluó el desempeño de diferentes modelos utilizando distintas técnicas de codificación de atributos categóricos y se compararon los resultados obtenidos para cuantificar el impacto de los codificadores en los modelos de ML. Finalmente, el segundo objetivo de este trabajo de investigación es generalizar los resultados obtenidos mediante una metodología que permita evaluar con fundamentos estadísticos las distintas técnicas de codificación.

Title

Methodology for representing categorical attributes with pattern-preserving codes and its application in machine learning methods.

Eric Valdez Valenzuela
Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México

Summary

The main goal of this research is focused on how to handle categorical attributes present in databases for their application in Machine Learning (ML) algorithms. Categorical attributes are those that contain a finite number of distinct values or groups. However, most ML algorithms, in practice, exclusively work with numerical attributes, so it is mandatory to process categorical attributes to represent them in a metric space. Although there are multiple techniques for converting categorical attributes to numerical values, if not applied correctly, existing patterns in categorical attributes may not be preserved. This loss of information affects the performance of ML models, making it crucial to retain the patterns embedded in categorical variables.

The information contained in categorical attributes can be represented through Pattern Preserving Codes (PPC). There are encoding techniques that find these PPCs, one of which is CESAMO: Categorical Encoding By Statistical Applied Modeling. This research aims to study and expand the knowledge of this technique. One of the main objectives is to determine how different encoding techniques affect ML algorithms. To achieve this, the performance of different models was evaluated using various categorical encoding techniques, and the results were compared to quantify the impact of the encoders on the ML models. Finally, the second objective of this research is to generalize the results obtained by developing a methodology that allows for the evaluation of different encoding techniques based on statistical foundations.

Capítulo 1

Introducción

1.1. Motivación

Dada la naturaleza del mundo que nos rodea, existen medidas o características de entidades que solo se pueden expresar con un número finito de categorías. Por ejemplo, el lugar de nacimiento de una persona, color de ojos, género, estado civil, tipo de vivienda, etc., son atributos cuyos posibles valores se representan con símbolos o palabras y tienen un número limitado de posibles opciones. Este tipo de atributos se encuentran en la mayoría de las bases de datos que se emplean para entrenar modelos de aprendizaje automático (ML por sus siglas en inglés). Denominamos bases de datos mixtas a aquellas que contienen tanto atributos categóricos como numéricos. Los atributos categóricos requieren un preprocesamiento que permita representarlos en un espacio métrico, es decir, asignarles valores numéricos. En términos matemáticos, un espacio métrico es un conjunto asociado a una función de distancia que satisface ciertas propiedades, de manera que, para cualquier par de puntos en el conjunto, esta función asigna una distancia que cumple dichas propiedades.

A pesar de la importancia de este preprocesamiento, esta área de la inteligencia artificial ha recibido poca atención. Las técnicas más comunes para abordar esta tarea, si no se emplean adecuadamente, pueden: 1) introducir información inexistente en los datos originales o 2) eliminar posibles patrones subyacentes en los valores categóricos. Estas dos consecuencias no deseadas afectan el desempeño de las técnicas de ML.

Por otro lado, existen algoritmos que permiten encontrar Códigos que Preservan Patrones para los atributos categóricos. Uno de ellos es CESAMO: *Categorical Encoding By Statistical Applied Modelling*. Este algoritmo se basa en el análisis estadístico de códigos numéricos y emplea una función de aproximación polinómica para encontrar posibles relaciones entre los atributos de un conjunto de datos. Se ha encontrado en trabajos previos que este enfoque obtiene resultados positivos en el desempeño de modelos de ML para tareas de clasificación y regresión. Se propone entonces explorar y ampliar los fundamentos de CESAMO para identificar áreas de posibles mejoras y, mediante una investigación de doctorado, desarrollar una metodología que facilite el preprocesamiento de bases de datos mixtas para su implementación en técnicas de aprendizaje automático.

1.2. Objetivos

El objetivo general de este trabajo consiste en diseñar y desarrollar un algoritmo que permita encontrar códigos numéricos que preservan patrones para los atributos categóricos presentes en las bases de datos mixtas y medir su efectividad mediante la comparación con otras técnicas. Además, se busca generalizar los resultados obtenidos mediante una metodología que permita evaluar con fundamentos estadísticos las distintas técnicas de codificación.

Los objetivos particulares consisten en:

- Analizar las áreas de oportunidad de las técnicas que procesan los atributos categóricos en Códigos que Preservan Patrones (PPCs por sus siglas en inglés).

- Diseñar y desarrollar una propuesta de un algoritmo basado en análisis estadístico para encontrar PPCs.
- Evaluar la propuesta para cuantificar su efecto en las técnicas de aprendizaje automático.
- Proponer mejoras al algoritmo con base en los resultados de manera iterativa.
- Formalizar los resultados mediante una evaluación basada en fundamentos estadísticos.

1.3. Hipótesis

Es factible representar atributos categóricos en un espacio métrico mediante Códigos que Preservan Patrones (PPC) a partir del estudio y optimización de un método que se base en el análisis estadístico. Se espera que, tras aplicar estos PPC, exista una mejora en el desempeño en las técnicas de aprendizaje automático en tareas de clasificación y regresión.

1.4. Preguntas de investigación

Las preguntas de investigación que se buscan responder en este trabajo son:

1. ¿Cómo afectan las técnicas de codificación de atributos categóricos a los modelos de aprendizaje automático?
2. ¿Es posible representar la información embebida en los atributos categóricos mediante códigos que preserven los patrones?

1.5. Contribución

Este trabajo de investigación de doctorado tiene dos principales contribuciones:

1. Desarrollar una técnica de codificación de atributos categóricos que encuentre los Códigos que Preservan los Patrones.

2. Evaluar y comparar el desempeño de diversas técnicas de codificación utilizando una metodología respaldada por fundamentos estadísticos.

1.6. Organización de la tesis

La presente tesis está estructurada en los siguientes capítulos:

En el capítulo 2 se proporciona un contexto histórico y conceptual sobre el tema de investigación, destacando los principales hitos, teorías y avances relevantes que han contribuido al desarrollo del campo de estudio.

En el capítulo 3 se presenta una revisión de la literatura actual relacionada con el tema de investigación, la cual incluye la descripción de las principales técnicas de codificación, y diversos estudios que han medido cómo estas influyen en los algoritmos de aprendizaje automático.

El capítulo 4 describe detalladamente la metodología propuesta para abordar el problema de investigación. Se explican los enfoques teóricos, técnicas y procedimientos utilizados para llevar a cabo el estudio, así como la justificación de su selección.

En el capítulo 5 se presentan y analizan los resultados obtenidos a partir de la aplicación de la metodología propuesta. Se incluyen gráficos, tablas y análisis estadísticos que respaldan las conclusiones de la investigación.

Finalmente, en el capítulo 6 se resumen las conclusiones principales derivadas de la investigación y se discuten sus implicaciones teóricas, prácticas y metodológicas. Además, se proponen áreas de investigación futura y posibles extensiones del estudio para continuar avanzando en el campo.

Capítulo 2

Antecedentes

2.1. Introducción

En este capítulo se presentan los fundamentos teóricos sobre la codificación de atributos categóricos y su uso en algoritmos de aprendizaje automático. Este marco conceptual permitirá al lector entender los desafíos asociados y la importancia de la codificación en el rendimiento de los modelos de aprendizaje automático. Además, proporcionará el contexto necesario para abordar la metodología propuesta en los capítulos siguientes.

2.2. Aprendizaje automático

El aprendizaje automático es un campo de la inteligencia artificial que se centra en el diseño y desarrollo de algoritmos que permiten a las computadoras mejorar su rendimiento en tareas específicas a través de la experiencia. Es especialmente útil en problemas donde no existen funciones o algoritmos bien definidos para su resolución, como la clasificación de datos complejos o la predicción de eventos futuros. Se basa en

la idea de que los sistemas pueden aprender de los datos, identificar patrones y hacer predicciones con un mínimo de intervención humana ([1]).

Los distintos algoritmos de aprendizaje automático se pueden clasificar en tres grupos principales, cada uno diseñado para resolver problemas de naturaleza distinta. Estos grupos son 1) el aprendizaje supervisado, 2) el no supervisado y 3) el aprendizaje por refuerzo. A continuación, se describen brevemente estos tres tipos de aprendizaje y sus aplicaciones más comunes.

2.2.1. Aprendizaje Supervisado

El aprendizaje supervisado emplea algoritmos que se entrenan utilizando un conjunto de datos etiquetado, es decir, los datos de entrenamiento contienen la variable que se busca predecir ([2]). En este enfoque, el modelo aprende a mapear entradas a salidas basándose en ejemplos previos proporcionados durante el entrenamiento. Los problemas que resuelve el aprendizaje supervisado se pueden agrupar en dos tipos:

- Regresión: busca predecir valores numéricos continuos a partir de las características de entrada del conjunto de datos. Algunos ejemplos de modelos que resuelven problemas de regresión son: Regresión Lineal, Regresión Polinómica y las Redes Neuronales Perceptrón Multicapa.

- Clasificación: Se enfoca en predecir si una instancia o registro en un conjunto de datos pertenece (o no) a una categoría específica, basándose en las propiedades de esa instancia. Esta clasificación puede ser binaria, donde la instancia se asigna a una de dos posibles categorías, o multiclase, donde la instancia puede pertenecer a una de varias categorías posibles. Los modelos comunes son: Regresión logística, las Máquinas de Soporte Vectorial (SVM), los Árboles de Decisiones y también las Redes Neuronales Perceptrón Multicapa.

Este tipo de aprendizaje tiene un impacto significativo tanto en la industria como en la academia. En el ámbito industrial, se emplea en problemas como la clasificación de correos electrónicos, la predicción de ventas en empresas minoristas, el diagnóstico médico

basado en imágenes o registros clínicos, y la optimización de procesos en manufactura. En la academia, se utiliza para explorar fenómenos complejos, como el análisis del comportamiento humano en psicología, el reconocimiento de patrones en bioinformática, la clasificación de galaxias en astronomía y la predicción de resultados experimentales en ciencias materiales. Su versatilidad lo convierte en una herramienta fundamental para resolver problemas en diversos dominios.

2.2.2. Aprendizaje No Supervisado

En contraste, en el aprendizaje no supervisado, el modelo se entrena utilizando conjuntos de datos no etiquetados. A diferencia del aprendizaje supervisado, en el que se conoce la variable a predecir, en el aprendizaje no supervisado es el modelo el que debe encontrar patrones, estructuras o relaciones en los datos ([3]). Este tipo de aprendizaje es particularmente útil cuando se desconoce la variable a predecir, o cuando se desea explorar los datos sin asumir una estructura predefinida.

Entre los algoritmos más comunes de aprendizaje no supervisado se encuentran:

- *Clustering*: Este conjunto de algoritmos organiza instancias similares en *clusters* o grupos. Uno de los algoritmos más usados es el de *K-means*, el cual es un método de *clustering* en el que los datos se dividen en K grupos, asignando cada instancia a un grupo con la media más cercana. El algoritmo ajusta iterativamente las posiciones de las medias de los grupos hasta que las asignaciones de instancias se estabilizan, minimizando la variación dentro de cada grupo ([4]).

- Reducción de Dimensionalidad: Estos algoritmos se utilizan para reducir el número de características en los datos, preservando la mayor cantidad posible de información relevante. Un ejemplo es el Análisis de Componentes Principales (*Principal Components Analysis* o PCA), el cual es un método que transforma un conjunto de datos de alta dimensión en un espacio de menor dimensión, conservando la mayor cantidad posible de la varianza presente en los datos originales. PCA logra esto identificando las direcciones de los vectores en el espacio de características, llamadas componentes principales, en las que los datos varían más,

y proyecta los datos a lo largo de estas direcciones (5).

En resumen, el aprendizaje no supervisado se utiliza para resolver problemas donde la estructura subyacente de los datos no es conocida de antemano. Algunos ejemplos de problemas que puede resolver incluyen la segmentación de clientes en marketing, la detección de patrones en datos biométricos y la compresión de datos.

2.2.3. Aprendizaje por Refuerzo

El aprendizaje por refuerzo es un área del ML en el que un agente aprende a tomar decisiones mediante la interacción con su entorno. A diferencia del aprendizaje supervisado y no supervisado, en el aprendizaje por refuerzo el agente no recibe ejemplos de entrada con salidas correctas, sino que explora el entorno donde se encuentra y recibe retroalimentación en forma de recompensas o penalizaciones, según las acciones que realiza (6). El objetivo del agente es aprender a tomar decisiones que maximicen la utilidad o recompensa recibida a lo largo del tiempo.

Entre los algoritmos más conocidos en aprendizaje por refuerzo se encuentran:

- *Q-Learning*: es un algoritmo que permite a un agente aprender reglas, también llamadas políticas o *policies*, para tomar decisiones óptimas y con ello maximizar la recompensa acumulativa en un entorno determinado. El algoritmo funciona actualizando iterativamente una función Q, que estima el valor esperado de una acción en un estado dado, basándose en la recompensa recibida y la estimación futura de las recompensas. (7)

- *Deep Q-Networks (DQN)*: es una extensión del algoritmo *Q-learning* que utiliza redes neuronales profundas para aproximar la función Q, lo que permite al agente aprender políticas óptimas en entornos con espacios de estados relativamente grandes. *DQN* es efectivo en problemas donde el espacio de estados es tan grande que resulta poco práctico gestionar el espacio en memoria que demanda dicho problema. El uso de redes neuronales permite generalizar entre estados similares y con ello manejar entornos complejos, como aquellos encontrados en juegos y simulaciones. (8)

El aprendizaje por refuerzo es particularmente útil en problemas donde las decisiones se deben tomar en secuencia y las acciones tienen un impacto acumulativo a largo plazo. Se emplea con mayor frecuencia en áreas como los videojuegos, donde agentes aprenden a jugar y mejorar a través de la experiencia; y la robótica, donde robots aprenden a realizar tareas complejas mediante prueba y error.

Después de esta introducción a los fundamentos del aprendizaje automático y sus principales enfoques, se procederá a analizar cómo se gestionan y representan los datos utilizados para entrenar estos algoritmos. En general, estos datos suelen estar compuestos por una combinación de atributos numéricos y categóricos.

Esta tesis se centra principalmente en los atributos categóricos, que plantean desafíos particulares en términos de procesamiento y codificación. Para que los modelos de aprendizaje automático puedan aprovechar de manera efectiva la información que contienen, es fundamental realizar un preprocesamiento adecuado de estos atributos. En las secciones siguientes, se analizarán en detalle qué son los atributos categóricos, su importancia en el aprendizaje automático y las técnicas más utilizadas para su codificación.

2.3. Atributos categóricos

Los atributos categóricos son variables que se clasifican en categorías o grupos distintos, en los que cada categoría representa un posible valor que la variable puede tomar. Estos atributos se utilizan para representar información cualitativa, como el género, el estado civil o el tipo de ocupación (9).

Existen dos tipos principales de atributos categóricos: nominales y ordinales (10). Los atributos categóricos nominales representan categorías sin ningún orden inherente entre ellas. Por ejemplo, en un conjunto de datos que describe coches, el atributo “Marca” sería nominal, ya que las marcas de coches no tienen un orden natural.

Por otro lado, los atributos categóricos ordinales representan categorías con un or-

den o jerarquía específica entre ellas. Por ejemplo, en el mismo conjunto de datos de coches, el atributo "Estado de Conservación" sería ordinal, ya que las categorías como "Excelente", "Bueno", "Regular" y "Malo" tienen un orden natural que refleja la calidad o condición del coche.

En la Tabla 2.1, se presenta un ejemplo de un conjunto de datos que contiene tanto atributos categóricos nominales como ordinales. El atributo "Marca" representa un atributo categórico nominal, mientras que el atributo "Estado de Conservación" representa un atributo categórico ordinal.

Tabla 2.1: *Dataset* de Coches

ID	Millas	Precio ($)	Marca	Estado de Conservación
1	50000	15000	Toyota	Bueno
2	80000	12000	Honda	Excelente
3	60000	18000	Ford	Regular
4	70000	10000	Chevrolet	Malo
5	90000	9000	Toyota	Bueno

Dado que la mayoría de los modelos de aprendizaje automático operan con valores numéricos ([11]), los atributos categóricos requieren un tratamiento diferente al de los atributos numéricos para garantizar que la información que contienen se utilice de manera adecuada al entrenar estos modelos.

2.4. Codificación de atributos categóricos

A pesar de la importancia de los atributos categóricos en la descripción de entidades del mundo real, muchos algoritmos de aprendizaje automático están diseñados para trabajar exclusivamente con datos numéricos, y la presencia de atributos categóricos puede representar un desafío para su implementación.

Por ejemplo, modelos como Regresión Lineal, Regresión Logística, Máquinas de Soporte Vectorial y Redes Neuronales suelen requerir que todas las variables de entrada sean numéricas. Esto significa que, para utilizar estos modelos con conjuntos de datos que contienen atributos categóricos, es necesario realizar una codificación de estos atributos en representaciones numéricas.

Por otro lado, es importante mencionar que existen modelos de aprendizaje automático que pueden, en teoría, manejar de manera nativa atributos categóricos. A continuación se listan los modelos más comunes: (12):

- Árboles de decisión: son algoritmos que dividen el espacio de características en nodos de decisión basados en valores discretos (13). Por ejemplo, si un atributo es categórico, como *Color*, el árbol de decisión puede dividir los datos en ramas correspondientes a cada instancia de color (rojo, verde, azul, etc.). De manera similar, el algoritmo de *Random Forest* combina múltiples árboles de decisiones para realizar predicciones más precisas. Cada árbol en el bosque se construye utilizando una muestra aleatoria del conjunto de datos y una selección aleatoria de características, lo que permite manejar atributos categóricos.

- *Gradient Boosting Machines (GBM)*: en esencia, los algoritmos de GBM construyen un modelo de predicción compuesto de un conjunto de modelos de aprendizaje débil, que son generalmente árboles de decisión poco profundos (14). Dado que estos modelos están basados en árboles de decisión, también pueden manejar atributos categóricos.

- Naive Bayes: es un modelo probabilístico que asume independencia entre las características. En el caso de ser entrenado con atributos categóricos, Naive Bayes calcula la probabilidad de ocurrencia de cada clase dado un valor específico de la característica categórica, y utiliza estas probabilidades para realizar predicciones (15).

Sin embargo, los modelos anteriores son excepciones a la regla. La mayoría de los modelos requieren que los atributos estén en forma numérica. Además, cuando estos algoritmos se implementan mediante las bibliotecas de código abierto comúnmente utilizadas como *Scikit-learn* (16) y *XGBoost* (17), surge la necesidad de mapear las instancias categóricas en valores numéricos, ya que estas bibliotecas aceptan únicamente entradas numéricas. En resumen, los atributos categóricos deben ser codificados a valores numéricos para la mayoría de las implementaciones de algoritmos de aprendizaje automático.

La codificación de atributos categóricos tiene dos objetivos principales en el aprendizaje automático:

- Preservar la información contenida en los atributos categóricos. Al convertir estos atributos en valores numéricos que los modelos de aprendizaje automático puedan interpretar, es crucial que la codificación mantenga las relaciones y diferencias entre las categorías. Esto debe hacerse evitando tanto la pérdida de información como la introducción de información espuria que podría comprometer la exactitud del modelo.

- Mejorar el rendimiento del modelo. Una codificación adecuada puede facilitar el aprendizaje del modelo al reducir la complejidad y tamaño de los datos, lo cual se refleja generalmente en una mejora en la eficiencia computacional, reduciendo el tiempo de entrenamiento y los recursos necesarios para procesar grandes volúmenes de datos categóricos. Además, si la codificación conserva los posibles patrones subyacentes entre el atributo categórico y los demás atributos en el conjunto de datos, esto mejora el desempeño de los modelos. Por el contrario, si se modifica la información, el desempeño del modelo puede verse afectado negativamente.

La elección de la técnica de codificación adecuada influye directamente en el desempeño del modelo entrenado. Es importante resaltar que la preservación de la información contenida en los atributos categóricos es fundamental, ya que una representación incorrecta puede llevar a que los modelos no logren capturar patrones relevantes o, peor aún, introduzcan relaciones artificiales que distorsionen la interpretación de los resultados. Por esta razón, es importante analizar con claridad el concepto de "información" en el contexto de los atributos categóricos y su relación con la eficacia de los modelos de aprendizaje automático.

2.4.1. Definición de Información

En este trabajo, el término *información* se utiliza para referirse a la cantidad y relevancia significativa que aportan los atributos categóricos a los modelos de aprendizaje

automático. Este concepto se alinea con la noción de información descrita en la Teoría de la Información de Shannon, donde la información se mide como una reducción en la incertidumbre sobre el sistema o el fenómeno observado (18).

Matemáticamente, la información contenida en una variable puede cuantificarse mediante la entropía, definida como (2.1):

$$H(A) = -\sum_{a \in A} p(a) \log_2(p(a)), \tag{2.1}$$

donde A es un conjunto de valores categóricos y $p(a)$ es la probabilidad de que ocurra un valor específico a. En este contexto, la entropía $H(A)$ mide la incertidumbre asociada a los valores de los atributos categóricos y, por ende, la cantidad de información presente antes de su codificación.

Cuando se aplica una técnica de codificación, una parte de esta información puede perderse debido a la transformación de los datos originales en nuevas representaciones. La pérdida de información puede ocurrir cuando las relaciones intrínsecas o patrones relevantes en los atributos categóricos no se preservan completamente en el espacio codificado.

En este sentido, al mencionar pérdida de información durante este trabajo, nos referimos específicamente a una disminución en la capacidad de los datos codificados para conservar patrones o estructuras relevantes para el aprendizaje automático. Esta pérdida puede evaluarse de manera empírica mediante el desempeño de los modelos de aprendizaje automático al ser combinados con distintas técnicas de codificación. Si un modelo obtiene un rendimiento significativamente inferior tras la codificación, esto sugiere que la transformación ha eliminado información útil para la tarea de predicción.

Además, la pérdida de información también puede cuantificarse de manera explícita utilizando métricas basadas en teoría de la información, como la *información mutua*. Esta métrica mide cuánta información comparten dos variables (19) y, en este contexto, nos permite evaluar qué tanto del contenido informativo original de los atributos categóricos se conserva después de su codificación.

Matemáticamente, la información mutua entre dos variables aleatorias A y B se define como (2.2):

$$I(A;B) = \sum_{a \in A} \sum_{b \in B} p(a,b) \log \frac{p(a,b)}{p(a)p(b)}, \tag{2.2}$$

donde A representa los valores originales de un atributo categórico y B sus valores codificados. $p(a,b)$ es la distribución conjunta de A y B, mientras que $p(a)$ y $p(b)$ son sus distribuciones marginales. Si la información mutua es alta, significa que la codificación ha logrado preservar la estructura de la información presente en los atributos originales. Por el contrario, si $I(A;B)$ es baja, indica que la codificación ha introducido pérdidas significativas, reduciendo la capacidad de los modelos de aprendizaje automático para explotar patrones en los datos.

La información mutua también puede expresarse en términos de entropía (2.3):

$$I(A;B) = H(A) + H(B) - H(A,B), \tag{2.3}$$

donde $H(A)$ y $H(B)$ son las entropías individuales de los datos originales y los datos codificados, respectivamente, y $H(A,B)$ es la entropía conjunta. Esta formulación nos permite interpretar la información mutua como la reducción de incertidumbre en una variable cuando se conoce la otra.

En el contexto de esta tesis, una alternativa potencial y prometedora para evaluar la calidad de las codificaciones sería analizar la información mutua entre los atributos categóricos originales y sus representaciones codificadas. Un método de codificación que maximice $I(A;B)$ aseguraría que la transformación preserve la información esencial de los atributos originales. Esto permitiría no solo representar los datos de manera eficiente, sino también conservar la información relevante para mejorar el desempeño de los modelos de aprendizaje automático.

Además de considerar la cantidad de información desde la perspectiva de la entropía y la información mutua, es relevante analizar la información desde el punto de vista de la *Teoría de la Complejidad de Kolmogorov*. Esta teoría mide la complejidad de un objeto, como una cadena de datos, mediante la longitud del programa más corto que puede generarlo en un lenguaje de programación universal (20). En otras palabras, la *comple-*

jidad de Kolmogorov de un atributo categórico A, denotada como $K(A)$, corresponde a la descripción más breve necesaria para representar A.

Matemáticamente, se expresa como 2.4:

$$K(A) = \text{mín}\{|p| : U(p) = A\}, \tag{2.4}$$

donde p es el programa que genera A, U es una máquina de Turing universal, y $|p|$ representa la longitud del programa.

Desde la teoría de la información, la Complejidad de Kolmogorov puede interpretarse como una medida de la compresión de los atributos categóricos. Antes de la codificación, estos poseen una complejidad asociada a sus relaciones intrínsecas. Sin embargo, al aplicar una técnica de codificación, esta complejidad puede aumentar si se introducen redundancias innecesarias o disminuir si se pierden patrones relevantes. La Complejidad de Kolmogorov ofrece una perspectiva adicional para evaluar la eficiencia de las técnicas de codificación al analizar cómo transforman la representación mínima de los datos. Una alta complejidad podría indicar que la codificación genera representaciones redundantes o difíciles de interpretar, mientras que una baja complejidad, junto con un buen desempeño del modelo, sugeriría una codificación eficiente que conserva patrones relevantes.

En esta tesis, nos enfocaremos principalmente en la medición de la conservación de la información mediante el desempeño que muestren los modelos de aprendizaje automático al ser combinados con las distintas técnicas de codificación. Este enfoque permite evaluar de manera práctica cómo las transformaciones realizadas por los codificadores afectan la capacidad de los modelos para aprender patrones relevantes, proporcionando así una medida implícita de la preservación de la información contenida en los atributos categóricos originales.

2.4.2. Desafíos en la codificación

Entre los desafíos más relevantes al mapear un atributo categórico a un espacio métrico se encuentran: la alta cardinalidad, el manejo del orden implícito en los datos,

y la escalabilidad cuando se trabaja con grandes volúmenes de datos. A continuación, se explican estos desafíos en detalle.

1) Alta cardinalidad. Este problema surge cuando un atributo categórico tiene un número elevado de categorías distintas. Por elevado se puede entender miles o incluso millones de instancias. Por ejemplo, un atributo como 'código postal' o 'ID de producto' puede tener entre miles y millones de valores únicos. La alta cardinalidad presenta distintos retos. Uno de ellos es el sobreajuste, donde el modelo se sesga durante el entrenamiento a un patrón específico, perdiendo la capacidad de generalización. Además, la codificación de atributos con alta cardinalidad (dependiendo de la técnica de codificación a emplear) puede aumentar significativamente las dimensiones del conjunto de datos, lo que no solo afecta el rendimiento del modelo, sino que también incrementa la demanda de almacenamiento y procesamiento computacional.

2) Atributos categóricos ordinales. Este tipo de atributos presentan un desafío adicional debido a la importancia del orden entre sus categorías. A diferencia de los atributos nominales, donde no existe un orden inherente, los atributos ordinales requieren una codificación que preserve esta jerarquía entre categorías. Si se ignora el orden implícito, se pierde información valiosa o incluso se puede introducir un patrón inexistente.

3) Escalabilidad. Este desafío surge durante la codificación de grandes volúmenes de datos que contienen múltiples atributos categóricos. A medida que el tamaño del conjunto de datos aumenta, también lo hace la complejidad de la codificación, lo que puede llevar a un incremento en el tiempo de procesamiento y en los recursos computacionales necesarios. Como se verá más adelante, este problema puede afectar más a algunas técnicas de codificación que a otras.

Para abordar estos desafíos, es importante elegir técnicas de codificación que sean eficientes, escalen adecuadamente con el volumen de datos, y preserven correctamente la información subyacente. La manera en que las distintas técnicas de codificación manejan estos desafíos impacta directamente en el rendimiento de los modelos de aprendizaje automático.

En el siguiente capítulo se examinarán con mayor profundidad las distintas técnicas

de codificación de atributos categóricos, explorando cómo estas influyen en el desempeño de los modelos.

Capítulo 3

Estado del arte

3.1. Introducción

Los conjuntos de datos empleados para entrenar modelos de aprendizaje automático normalmente contienen atributos categóricos. A diferencia de los atributos numéricos, este tipo de datos no son métricos y, por lo tanto, no pueden ser procesados directamente por la mayoría de los algoritmos de aprendizaje automático.

Para que estos algoritmos puedan procesar la información contenida en estos atributos, es necesario mapear las instancias categóricas a un valor numérico. Este proceso de transformación se conoce como codificación de atributos categóricos. Existen diversas técnicas de codificación, cada una de ellas tiene sus ventajas y limitaciones.

En este capítulo se presenta el estado del arte en el área de las técnicas de codificación de atributos categóricos y su efecto en los distintos algoritmos de aprendizaje automático. Como se verá durante el transcurso de este trabajo de doctorado, las técnicas de codificación de atributos categóricos juegan un papel importante en el procesamiento de datos para los modelos de aprendizaje automático.

Estas técnicas de codificación se clasifican en diferentes grupos según su enfoque. Se pueden dividir en técnicas supervisadas y no supervisadas, dependiendo de si utilizan o no la información de la variable objetivo durante el proceso de codificación. En el resto del documento usaremos de manera intercambiable el término variable objetivo o variable dependiente, y nos referimos a aquella variable que se busca predecir en un problema supervisado (también llamado *target* en inglés).

Además, se introduce un enfoque novedoso para tratar atributos categóricos, el cual se basa en el uso de Códigos que Conservan los Patrones. Finalmente, se expondrán diversos trabajos que exploran cómo las distintas técnicas de codificación influyen en el desempeño de los algoritmos de aprendizaje automático.

3.2. Técnicas no supervisadas

Las técnicas de codificación no supervisadas, al igual que los modelos no supervisados, se caracterizan por no utilizar información sobre la variable objetivo durante el proceso de codificación. En cambio, estas técnicas se basan únicamente en la estructura de los datos de entrada, o variables independientes. A continuación, se presentan algunas de las técnicas de codificación no supervisada más frecuentemente utilizadas en la literatura.

3.2.1. Ordinal Encoding

La técnica *Ordinal Encoding* transforma las categorías en números consecutivos, donde cada categoría se representa de manera arbitraria por un número entero distinto ([21]). *Ordinal Encoding* ofrece varias ventajas. En primer lugar, es una técnica relativamente simple de implementar, lo que la hace adecuada para situaciones en las que se necesita una solución de bajo costo computacional. Además, no aumenta la dimensión del conjunto de datos como lo hacen otras técnicas, lo que es beneficioso en términos de eficiencia computacional y manejo de memoria. Otra ventaja es su compatibilidad con algoritmos que consideran el orden, lo que resulta especialmente útil cuando las categorías tienen un orden inherente en los datos, como en niveles educativos o esca-

las de evaluación. En estos casos, la codificación ordinal puede ser beneficiosa, ya que preserva la relación de orden entre las categorías.

Sin embargo, esta técnica también tiene desventajas importantes. Una de las principales es que puede introducir información espuria cuando no existe un orden natural en los datos, lo que puede llevar a sesgos en los modelos de aprendizaje automático. Esto es particularmente negativo en modelos de regresión y clasificación que asumen que las diferencias entre los valores numéricos son significativas, lo que puede resultar en interpretaciones incorrectas de las relaciones entre las categorías.

3.2.2. One-hot Encoding

Cuando se implementa *One-hot Encoding*, cada categoría se transforma en una nueva columna binaria en la que solo una posición puede contener un valor distinto de '0', y empleando, frecuentemente, pero no necesariamente, '1' para indicar la presencia de la categoría ([22]). De hecho, cualesquiera dos valores para indicar presencia o ausencia de una categoría son válidos. La frecuente elección de '0' y '1' es arbitraria. Esta técnica es especialmente útil cuando el atributo es nominal, es decir, no hay un orden inherente entre las categorías, ya que evita introducir un orden artificial entre ellas.

Un ejemplo de su funcionamiento sería el siguiente: si se tiene un atributo categórico 'Color' con tres categorías: 'Rojo', 'Verde', 'Azul', *One-Hot Encoding* agregará tres columnas nuevas al conjunto de datos. Para un registro o fila que originalmente contenía el valor 'Rojo', el vector correspondiente será [1, 0, 0], para 'Verde' será [0, 1, 0], y para 'Azul' será [0, 0, 1].

La principal ventaja de esta técnica es que evita introducir un orden artificial entre las categorías, lo cual es crucial cuando se trabaja con datos categóricos nominales. Esto es especialmente útil en algoritmos como las redes neuronales y los modelos de regresión logística.

No obstante, su principal desventaja es el incremento en la dimensión del conjunto de datos, sobre todo en atributos de alta cardinalidad, es decir, con muchas categorías distintas ([23]). Este problema, conocido como la maldición de la dimensionalidad (*curse of dimensionality*), puede afectar el rendimiento del modelo. Además, en algunos casos,

la interpretación de los resultados se vuelve más compleja debido al número de columnas generadas.

3.2.3. Binary Encoding

La técnica *Binary Encoding* convierte cada categoría en un número entero y luego lo transforma en su representación binaria. Esta representación binaria se descompone en columnas separadas, con cada bit del número binario ocupando una columna diferente (24). Por ejemplo, si un atributo tiene tres categorías 'A', 'B' y 'C', estas podrían asignarse a los números 1, 2 y 3, que se transformarían en '001', '010' y '011' en binario pesado, respectivamente. Estos valores binarios se distribuyen en 3 columnas nuevas, donde cada bit se representa en una columna individual, permitiendo que los datos categóricos se conviertan en un formato que los modelos de aprendizaje automático puedan procesar. En comparación con la técnica *One-hot encoding*, la cual para un atributo con d instancias categóricas distintas agregaría d columnas, la técnica de *Binary Encoding* solo requiere $\log_2(d)$ columnas para el binario pesado.

Una de las principales ventajas de esta técnica es que ayuda a reducir la dimensión del conjunto de datos en comparación con otras técnicas. En lugar de crear una columna por cada categoría, *Binary Encoding* genera menos columnas, lo que puede ser beneficioso cuando se trabaja con atributos de alta cardinalidad. Además, la representación numérica binaria resultante es compatible con la mayoría de los algoritmos de aprendizaje automático, lo que hace que esta técnica sea flexible y aplicable en una variedad de escenarios.

Sin embargo, *Binary Encoding* también tiene sus desventajas. Una de las más significativas es la pérdida de interpretación en las columnas generadas (*interpretability* en inglés). A diferencia del *One-Hot Encoding*, donde cada columna corresponde a una categoría específica, las columnas creadas por el *Binary Encoding* representan bits de un número binario, lo que no tiene un significado claro y directo en relación con las categorías originales. Esta falta de transparencia puede complicar la comprensión de cómo los datos categóricos están siendo utilizados por el modelo.

3.2.4. Count Encoding

Count Encoding convierte atributos categóricos en valores numéricos basados en la frecuencia de aparición de cada categoría en el conjunto de datos ([25]). En lugar de asignar un valor arbitrario o un código binario, el *Count Encoding* asigna a cada categoría un valor numérico que corresponde al número de veces que esa categoría aparece en el conjunto de datos. Por ejemplo, si en un conjunto de datos de colores, 'rojo' aparece 10 veces, 'azul' 5 veces y 'verde' 2 veces, estos valores se asignarán directamente como los valores codificados: 'rojo' se reemplaza por '10', 'azul' por '5', y 'verde' por '2'.

Una de las principales ventajas del *Count Encoding* es su simplicidad y eficiencia, especialmente cuando se trabaja con conjuntos de datos grandes o con atributos categóricos de alta cardinalidad. Como no introduce nuevas dimensiones en el conjunto de datos, el *Count Encoding* es más eficiente en términos de espacio y tiempo que otras técnicas como el *One-Hot Encoding*. Además, al preservar la frecuencia de las categorías, puede ayudar a capturar cierta información estadística sobre la distribución de los datos que podría ser útil para algunos modelos de aprendizaje automático.

Por otro lado, una de sus desventajas más significativas es que puede introducir un sesgo en los modelos de aprendizaje automático, particularmente si algunas categorías son mucho más frecuentes que otras. Esto implica que algunos modelos asignen más importancia a las categorías más comunes, independientemente de su relevancia real en la predicción.

3.2.5. Hashing Encoding

A diferencia de otras técnicas de codificación que asignan un valor específico a cada categoría, el *Hashing Encoder* aplica una función de *hashing* que transforma las categorías en un rango predefinido de números ([26]). Esta técnica no requiere almacenar un mapa de correspondencia entre cada categoría y el valor codificado, lo que la hace altamente escalable y eficiente en términos de memoria. Por ejemplo, si se tiene un conjunto de datos con las categorías 'gato', 'perro' y 'pájaro', el *Hashing Encoding* aplicará una

función de *hash* a cada una, generando valores numéricos que luego se pueden utilizar como entrada para los modelos de aprendizaje automático.

La principal ventaja *Hashing Encoding* se encuentra en su eficiencia y escalabilidad, lo que lo hace especialmente útil en conjuntos de datos donde el número de categorías es muy grande y podría contener instancias anteriormente no vistas por el codificador. Debido a que no requiere un diccionario de categorías, el *Hashing Encoding* es menos susceptible a problemas de memoria y puede manejar dinámicamente nuevas categorías que no estaban presentes en los datos de entrenamiento. Esta flexibilidad es una gran ventaja cuando se trabaja con datos en tiempo real o cuando se procesan flujos continuos de información.

No obstante, su mayor desventaja (como ocurre siempre que se usa una función de hash) es la posibilidad de colisiones, donde diferentes categorías se mapean al mismo valor numérico debido a la naturaleza de la función de *hashing*. Esto puede introducir ruido en los datos y degradar el rendimiento del modelo. Además, al ser un proceso basado en *hashing*, la codificación no es invertible; es decir, no se puede recuperar la categoría original a partir del valor *hasheado*, lo que puede complicar la interpretación y el análisis de los resultados.

3.3. Técnicas supervisadas

A diferencia de las técnicas no supervisadas, las técnicas de codificación supervisadas hacen uso de la variable objetivo durante el proceso de codificación. Esto permite que la codificación de los atributos categóricos capture información relevante de la posible relación existente entre el atributo categórico y la variable a predecir. Sin embargo, es importante destacar que estas técnicas están limitadas a problemas de aprendizaje supervisado, ya que requieren de la disponibilidad de la variable objetivo para realizar la codificación. A continuación, se describen algunas de las técnicas de codificación supervisadas más comunes en el ámbito de la computación.

3.3.1. Target Encoding

Target Encoding es una técnica de codificación supervisada que transforma atributos categóricos en valores numéricos basándose en la relación entre cada categoría y la variable a predecir (27). Esta técnica asigna a cada categoría un valor que refleja la media o promedio de la variable objetivo para esa categoría. A continuación se describe cómo se puede implementar esta técnica en un problema de clasificación binaria. Primero es necesario determinar, para cada instancia categórica c, el promedio o media de veces que la instancia tuvo un valor positivo ($n+$) en la variable objetivo, considerando todas sus apariciones en el atributo categórico (n). Esto se denota como Media_c y se describe en la Ecuación 3.1

$$\text{Media}_c = \frac{n+}{n} \tag{3.1}$$

Posteriormente, se requiere determinar la media global de ocurrencias positivas en la variable objetivo. Como se muestra en la Ecuación 3.2, esto se obtiene dividiendo el número total de valores positivos $y+$ entre el número total de instancias en la variable objetivo y,

$$\text{Media Global} = \frac{y+}{y} \tag{3.2}$$

Para evitar el sobreajuste en casos donde algunas categorías tienen pocas observaciones, se puede aplicar un proceso de ponderación. La Ecuación 3.3 describe cómo se calcula la Media Ponderada para una categoría c. En esta ecuación, n_c representa el número de instancias en la categoría c, mientras que λ es un parámetro de ponderación que regula la influencia de la media global en la media final.

$$\text{Media Ponderada}_c = \frac{\text{Media}_c \cdot n_c + \text{Media Global} \cdot \lambda}{n_c + \lambda} \tag{3.3}$$

Finalmente, el valor codificado para una instancia se obtiene asignando a dicha instancia la Media Ponderada correspondiente a su categoría c. Este valor codificado reemplaza a la categoría original en el conjunto de datos.

Una de las principales ventajas del *Target Encoding* es que aprovecha la información de la variable objetivo para codificar las categorías, lo que puede mejorar la capacidad de predicción del modelo al capturar de cierta manera la relación entre las categorías y la variable objetivo. Esta técnica puede ser especialmente efectiva en problemas donde la variable categórica tiene una fuerte influencia en la variable objetivo. Además, *Target Encoding* convierte las categorías en un solo valor numérico en lugar de crear múltiples columnas.

A pesar de sus beneficios, el *Target Encoding* también presenta algunas desventajas. La principal es el riesgo de sobreajuste del modelo, especialmente si el conjunto de datos es pequeño o tiene muchas categorías con pocas observaciones. La codificación basada en la media de la variable objetivo puede hacer que el modelo se ajuste demasiado a las particularidades del conjunto de entrenamiento, afectando su capacidad de generalización.

Además, es importante mencionar el riesgo de *target leakage* o filtración del objetivo, el cual ocurre cuando la codificación de los atributos categóricos incluye información de la variable objetivo que, en un escenario real, no estaría disponible en el momento de realizar las predicciones ([28]). Esto puede suceder, por ejemplo, si se calcula la media de la variable objetivo para cada categoría utilizando todo el conjunto de datos, incluidos los registros de prueba. En este caso, la técnica *Target Encoding* utiliza información que pertenece a los datos de evaluación o prueba, comprometiendo el principio de independencia entre los conjuntos de entrenamiento y prueba.

El principal impacto del *target leakage* es que conduce a un sobreajuste del modelo, ya que este aprende patrones que no están disponibles en un entorno de producción. Esto resulta en un buen desempeño del modelo durante la validación cruzada o pruebas internas, pero genera un rendimiento deficiente cuando se aplica a datos completamente nuevos. En otras palabras, las métricas de evaluación del modelo dejan de ser representativas de su capacidad real para generalizar.

Además, el *Target Encoding* puede no ser tan efectivo en situaciones donde la variable objetivo no tiene una relación clara con las categorías, o en problemas de clasificación con un alto número de categorías distintas.

Otras técnicas de codificación supervisada, como *MEstimate*, *Leave One Out*, *James-Stein* y *CatBoost Encoder*, comparten fundamentos con el *Target Encoding*, pero introducen variaciones específicas para abordar sus limitaciones.

Entre estas variantes, el *CatBoost Encoding* se destaca por su creciente popularidad y eficacia en la mitigación del *Target Leakage*, lo que mejora notablemente la capacidad de generalización de los modelos ([29]). A continuación, se detalla el funcionamiento del *CatBoost Encoder*.

3.3.2. CatBoost Encoder

El *CatBoost Encoder* es una técnica de codificación supervisada que, al igual que el *Target Encoder*, transforma atributos categóricos en valores numéricos basándose en la relación entre cada categoría y la variable a predecir. Sin embargo, a diferencia del *Target Encoder*, esta técnica introduce el concepto de codificación progresiva, el cual reduce significativamente el riesgo de sobreajuste y minimiza el efecto de *target leakage*. Esta técnica es particularmente efectiva cuando se emplea con modelos basados en árboles de decisión, como *XGBoost* y *LightGBM*.

CatBoost Encoder codifica cada categoría utilizando la media acumulada de la variable objetivo de todas las instancias anteriores en el conjunto de datos, mientras que la primera instancia de cada categoría se codifica utilizando la media global. Esto tiene como propósito evitar que el codificador tenga acceso a toda la información de la variable objetivo que no debería estar disponible en el momento de realizar una codificación para una instancia categórica en particular ([30]).

A continuación, se describe cómo se puede implementar esta técnica en un problema de clasificación binaria:

Cálculo de la Media Progresiva

Para cada categoría c y en cada instancia i, calculamos la media acumulada de la variable objetivo (y) de todas las instancias anteriores que pertenecen a la misma

categoría c. Este valor se denota como Media Progresiva$_{c,i}$ y se calcula como (Ecuación 3.4) :

$$\text{Media Progresiva}_{c,i} = \frac{\sum_{j=1}^{i-1} y_j}{n_c^{(i-1)}} \tag{3.4}$$

Donde:

- y_j : es el valor correspondiente de la variable objetivo para la instancia j, cuando la categoría es c.
- $n_c^{(i-1)}$:es el número de instancias anteriores a i que pertenecen a la categoría c.

Si $i = 1$ (es decir, la primera instancia de la categoría c), no hay valores anteriores para promediar, por lo que se utiliza la Media Global de la variable objetivo en su lugar.

Esta Media Global se calcula de la misma manera que en el *Target Encoder* (Ecuación 3.2).

Finalmente, el valor codificado para la categoría c en la instancia i se obtiene utilizando la media progresiva o la media global, dependiendo de si es la primera instancia de esa categoría o no:

$$\text{Valor Codificado}_{c,i} = \begin{cases} \text{Media Global} & \text{si } i = 1 \\ \text{Media Progresiva}_{c,i} & \text{si } i > 1 \end{cases} \tag{3.5}$$

Esta codificación progresiva permite que el *CatBoost Encoder* capture la información relevante de las categorías sin introducir el riesgo de *target leakage*.

CatBoost Encoder es particularmente útil en situaciones donde las categorías tienen un número reducido de instancias, ya que reduce el riesgo de sobre ajuste al no permitir que el modelo obtenga información de la variable objetivo de una instancia específica durante el proceso de codificación (siempre se basa en instancias anteriores para calcular el valor de la instancia actual).

En resumen, *CatBoost Encoder* es una extensión del *Target Encoder* que mejora su robustez y capacidad de generalización mediante el concepto de 'codificación progresiva'

3.4. Técnicas que emplean Códigos que Conservan los Patrones

Además de las técnicas de codificación convencionales, existe un enfoque que busca preservar los posibles patrones inherentes en todos los atributos presentes en el conjunto de datos mediante Códigos que Conservan los Patrones (31), o *Pattern Preserving Codes (PCCs)* por sus siglas en inglés.

Las técnicas basadas en PPCs tienen como objetivo principal mantener las relaciones subyacentes entre un atributo categórico y los demás atributos en el conjunto de datos al transformarlos en valores numéricos. Mediante el uso de PPCs, estas técnicas minimizan la pérdida de información y mejoran el rendimiento de los modelos de aprendizaje automático.

Los PPCs se pueden obtener de la siguiente manera. Consideremos un conjunto de tuplas de n-dimensiones, denotado como U, que contiene m elementos. Supongamos que existen n funciones desconocidas de $n-1$ variables, representadas de la siguiente manera:

$$f_k(v_1, ..., v_{k-1}, v_{k+1}, ..., v_n); \quad k = 1, ..., n \tag{3.6}$$

Supongamos también que existe un método que nos permite aproximar una función desconocida f_k con una función conocida F_k. Las funciones de aproximación resultantes de $n-1$ variables independientes se expresan como:

$$F_k \approx f_k(v_1, ..., v_{k-1}, v_{k+1}, ..., v_n); \quad k = 1, ..., n \tag{3.7}$$

La diferencia entre f_k y F_k se denota por e_k, que representa el error entre la función aproximada y la función real para el atributo k en las m tuplas del conjunto de datos:

$$e_k = \text{máx}\left(|f_{k,i} - F_{k,i}|\right); \quad i = 1, ..., m \tag{3.8}$$

El objetivo de los Códigos que Conservan los Patrones es minimizar e_k para todos los k, lo que garantiza que las relaciones entre la variable k y las demás $n-1$ varia-

bles se preserven en todo el conjunto de datos. Esto se formula como un problema de optimización multiobjetivo, expresado de la siguiente manera:

$$N = \text{mín}\left[\text{máx}\left(e_k; \quad k = 1, ..., n\right)\right] \tag{3.9}$$

Esta minimización asegura que solo aquellos códigos que mantengan las relaciones entre todas las variables en el conjunto de datos preservarán los patrones presentes en la base de datos.

Es importante notar que este es un problema de optimización multiobjetivo, ya que cumplir con la condición k en la ecuación 3.8 para un valor dado de k puede implicar el desajuste de la misma condición para un valor diferente de k. El uso de la expresión *min-max* en la ecuación 3.9 equivale a seleccionar un punto particular en el frente de Pareto ([32]).

En la literatura se encuentran al menos dos técnicas de codificación de atributos categóricos que utilizan estos Códigos que Conservan los Patrones. A continuación, se presentan ambas.

3.4.1. CENG

La técnica de codificación CENG (por sus siglas en inglés *Categorical Encoding with Neural Networks and Genetic Algorithms*) ([33]) convierte las instancias categóricas de un conjunto de datos en Códigos que Conservan los Patrones (PPCs). Para encontrar estos códigos, CENG resuelve dos problemas principalmente.

- 1) la técnica busca preservar la estructura presente en las relaciones entre las variables categóricas y numéricas del conjunto de datos. Esto se logra entrenando una red neuronal de perceptrón multicapa (MLP), que se utiliza para aproximar una función multivariable $f_i(n-1)$, que describe una variable como función del resto de las variables $n-1$. El error de aproximación máximo e_i asociado con cada variable es minimizado para reflejar el ajuste global de los códigos. El objetivo es encontrar los códigos que minimicen el error global $e_k = \text{máx}(e_i)$.

- 2) CENG utiliza un algoritmo genético (GA) para identificar el mejor conjunto de códigos numéricos a partir de una gran cantidad de combinaciones posibles. Cada individuo en el GA corresponde a un conjunto de números binarios, y la aptitud (*fitness*) de cada individuo está determinada por e_k. El GA minimiza este error, generando una aproximación precisa de los códigos numéricos que preservan los patrones

Al resolver estos dos problemas, CENG garantiza que los códigos resultantes permitan expresar cada variable como una función de las restantes, minimizando el error de predicción y asegurando la preservación de la estructura de los datos. Estos códigos numéricos permiten aplicar métodos numéricos convencionales a las bases de datos con atributos categóricos, preservando los patrones inherentes en los datos originales.

3.4.2. CESAMO

Otra técnica importante para encontrar PCCs, es CESAMO: *Categorical Encoding by Statistical Applied Modeling* (34). CESAMO está diseñado para reducir los altos costos computacionales asociados a técnicas previas como CENG. Esta técnica se basa en métodos estadísticos y numéricos, como alternativa al uso de redes neuronales o algoritmos genéticos, logrando resultados similares de forma más eficiente. El objetivo principal de CESAMO es asignar PPCs a los atributos categóricos de manera que se preserven los patrones de comportamiento que subyacen en los datos. A continuación se describe el algoritmo CESAMO:

1. Especificar la base de datos mixta (MD).

2. Especificar el tamaño de la muestra (ss).

3. Analizar la base de datos MD para determinar: el total de atributos numéricos n, el número de tuplas t, y el número de atributos categóricos c_i.

4. Para cada variable categórica v_i, seleccionar una variable v_j al azar y asignar valores aleatorios a las instancias de v_i.

5. Repetir hasta que la media y_i^{AVG} siga una distribución normal.

6. Calcular $y_i = f(v_j)$ y determinar los mejores códigos para v_i.

El algoritmo de CESAMO presenta 2 grandes desafíos: a) Cómo definir la función que preservará los patrones, y b) Cómo determinar el número de códigos a muestrear.

Respecto al desafío a), se selecciona como función de aproximación un polinomio de grado 11 (3.10)

$$y_i \leftarrow P_{11}(x) \sim \beta_0 + \sum_{i=1}^{6} \beta_i x^{2i-1} \tag{3.10}$$

En el trabajo (35) se mostró que cualquier función continua puede ser aproximada con una combinación lineal de monomios con términos de grado impar. Esta función polinómica conserva relaciones lineales y no lineales.

Respecto al desafío b), se conoce por el Teorema del Límite Central que, independientemente de la distribución de los yi, la distribución de las medias de las muestras de yi (yi_{AVG}) se volverá una distribución Gaussiana o normal. Una vez que la distribución de yi_{AVG} se vuelve normal, se logra estabilidad estadística, en el sentido de que un muestreo adicional de los yi no modificará significativamente la caracterización de la población.

En esencia, CESAMO propone muestrear suficientes códigos para garantizar la estabilidad estadística de los valores calculados a partir de $yi \leftarrow f(vj)$. Si $f(vj)$ se elige adecuadamente, los PPCs correspondientes a la mejor aproximación reemplazarán a las instancias categóricas. Además, CESAMO se basa en un muestreo de doble nivel: solo se consideran pares de variables en cada iteración del algoritmo. Sin embargo, al aproximar a la variable categórica con otra al azar en cada iteración, se está muestreando el espacio de todas las variables. Esto evita la necesidad de resolver explícitamente el problema de optimización multiobjetivo subyacente.

3.5. Efecto de las técnicas de codificación en algoritmos de Aprendizaje Automático

Independientemente del enfoque que empleen, todas las técnicas de codificación tienen distintos efectos en los modelos de aprendizaje automático. Existen diversos trabajos que han analizado y cuantificado cómo las técnicas de codificación influyen en el desempeño de distintos modelos. A continuación, se citan los más relevantes.

- En (36), los autores comparan 9 técnicas de codificación, empleando conjuntos de datos con atributos categóricos de alta cardinalidad. En el marco comparativo se incluyeron 5 diferentes algoritmos de aprendizaje supervisado: regresión LASSO, bosques aleatorios, *gradient boosting*, *k-nearest neighbors* y máquinas de soporte vectorial. Estos algoritmos se utilizaron para resolver 27 problemas de regresión, clasificación binaria y clasificación multiclase. Los autores encontraron que la técnica con el mejor desempeño para la mayoría de los modelos fue la de *Target Encoding* regularizada.

- De manera similar, en el artículo (37) se explora cómo distintas técnicas de codificación afectan el rendimiento de los modelos de aprendizaje automático. El estudio evalúa 14 técnicas de codificación, incluidos *one-hot* y *target encoding*, en 8 modelos de aprendizaje automático, mientras resolvieron 28 problemas de clasificación y regresión. Los autores encontraron que las redes neuronales multicapa obtuvieron los mejores desempeños con la técnica *one-hot encoding*. Por otro lado, también encontraron que la técnica *target encoding* y sus variantes obtuvieron los mejores resultados cuando se emplearon en modelos basados en árboles de decisiones.

- En (38), los autores presentan lo que ellos denominan el marco comparativo más completo hasta su publicación (2023). En su trabajo abordan limitaciones de marcos comparativos anteriores y evalúan 32 técnicas de codificación en 50 conjuntos de datos. Sus hallazgos muestran que el codificador supervisado *Weight of Evidence* logró los mejores desempeños con los árboles de decisión, mientras que el

modelo de regresión logística obtuvo los mejores resultados con los codificadores *Sum*, *One-Hot*, *Binary*, y *Weight of Evidence*.

- En el trabajo (39), los autores también desarrollaron un marco comparativo compuesto de 16 técnicas de codificación evaluadas en 15 conjuntos de datos y utilizando 7 modelos de aprendizaje automático. De acuerdo a sus resultados, los codificadores de propósito general con los mejores rendimientos fueron *Catboost*, *LeaveOneOut* y *Target encoder*.

- En un estudio más acotado (40), los autores midieron el desempeño de una red neuronal perceptrón multicapa al resolver un problema de clasificación (*Car Evaluation Data Set*). Los autores emplearon 7 técnicas de codificación distintas, y se compararon los desempeños obtenidos por la red neuronal con cada técnica de codificación. Se encontró que las técnicas con los mejores desempeños fueron *Sum encoding* y *Backward Difference Coding*

- Por último, el estudio (41) introduce una amplia e interesante recopilación de técnicas disponibles para procesar atributos categóricos en algoritmos de aprendizaje profundo (redes neuronales perceptrón multicapa), un tema que ha ganado amplia relevancia recientemente debido al éxito que han tenido las redes neuronales en Inteligencia Artificial Generativa. Los autores clasifican a las técnicas de codificación en tres grupos:

    - Deterministas: tienen la característica de que, dado el mismo conjunto de datos, siempre se obtendrán la misma conversión entre una instancia categórica y un valor numérico. Además, tiene una relativamente baja complejidad computacional. Ejemplos: *One-hot encoding*, *Ordinal encoding*

    - Algorítmicas: este tipo de técnicas pueden o no tener resultados deterministas. Se distinguen de las deterministas porque son más complejas en términos de procesamiento computacional. Otra distinción de este grupo, es que el procesamiento del atributo categórico se realiza antes de la etapa de entrenamiento del algoritmo de aprendizaje profundo. Ejemplos: *Latent Dirichlet*

Allocation.

- Automáticas: para este grupo, se emplean redes neuronales para generar dinámicamente representaciones de los valores categóricos como parte de la fase de entrenamiento. En otras palabras, este tipo de técnicas codifican los atributos categóricos como un efecto secundario durante la etapa de entrenamiento de una red neuronal. Estas técnicas son mejor conocidas como *embeddings*.

El estudio del estado del arte ha permitido identificar las fortalezas y limitaciones de las distintas técnicas de codificación de atributos categóricos, así como su impacto en los modelos de aprendizaje automático. Con base en estos antecedentes, en el siguiente capítulo se presenta la metodología propuesta en esta investigación.

Capítulo 4

Metodología propuesta

4.1. Introducción

En este capítulo se presenta la metodología implementada para abordar los principales objetivos de la investigación. La metodología global se divide en tres etapas, cada una enfocada en resolver un objetivo particular del proyecto de investigación. Es importante mencionar que el resultado de cada etapa determinó la dirección de la siguiente. Las etapas, ordenadas cronológicamente, se resumen de la siguiente manera:

- Etapa I: Estudio y cuantificación del efecto de las distintas técnicas de codificación en los modelos de aprendizaje automático.
- Etapa II: Diseño y desarrollo de una propuesta de un codificador basado en CESAMO.
- Etapa III: Implementación de una evaluación respaldada estadísticamente para comparar el desempeño de las técnicas de codificación.

Las siguientes secciones describen la metodología empleada en cada etapa.

4.2. Etapa I: Desarrollo de un Marco Comparativo

La metodología descrita en esta sección se encuentra publicada en el artículo 'Measuring the Effect of Categorical Encoders in Machine Learning Tasks Using Synthetic Data' [(42)] .

El objetivo de la primera etapa de la investigación se enfocó en responder la siguiente pregunta: *¿Cómo afectan las técnicas de codificación a los algoritmos de aprendizaje automático?*. Si bien actualmente (año 2025) existen diversos estudios que han hecho aportes importantes a esta pregunta, durante el tiempo de esta primera etapa (año 2020), existían pocos trabajos al respecto, por lo que se propuso desarrollar un Marco Comparativo (*benchmark*).

4.2.1. Fundamentos y justificación

El principal objetivo de este marco comparativo fue cuantificar y comparar el desempeño de distintas técnicas de codificación durante su implementación en modelos de aprendizaje automático. Para su desarrollo se propuso evaluar 10 técnicas de codificación, donde cada técnica sería combinada con 5 modelos, de tal manera que se obtuvieron 50 combinaciones de modelos y codificadores. A su vez cada combinación de modelo-codificador fue evaluado con 7 conjuntos de datos del mundo real, mismos que fueron tratados como problemas de clasificación y regresión. Además de los conjuntos de datos del mundo real, también se emplearon datos sintéticos, esto con el fin de cuantificar de manera más amplia el rendimiento de cada combinación.

Para cada modelo, se registró cuál codificador obtuvo el rendimiento más alto y más bajo, así como la diferencia entre estas dos mediciones.

En las siguientes secciones se describen las 3 partes principales que componen al marco comparativo: los modelos de aprendizaje automático, las técnicas de codificación y los conjuntos de datos.

4.2.2. Modelos evaluados

El marco comparativo incluyó 5 algoritmos de aprendizaje automático, junto con 10 técnicas de codificación. Para distinguir cada combinación de modelo-codificador, se asignaron abreviaturas que simbolizan los nombres de cada modelo y codificador respectivamente. Los algoritmos de aprendizaje automático utilizados junto con sus códigos se enumeran en la Tabla 4.1. Las abreviaturas se derivan del nombre de los algoritmos en inglés

Algoritmos	**Abreviatura**
Regresión Logística	LR
Gaussian Naïve-Bayes	GNB
Máquina de Soporte Vectorial	SVM
Red Neuronal (perceptrón multicapa)	NN
XGBoost	XGB
Bosques Aleatorios	RF

Tabla 4.1: Algoritmos de aprendizaje automático incluidos en el estudio comparativo

Para la implementación de los algoritmos de regresión logística (LR), Gaussian Naïve-Bayes (GNB), Máquina de Soporte Vectorial (SVM) y Bosques Aleatorios (RF) se utilizó la biblioteca de Scikit Learn (16). Todos estos algoritmos fueron entrenados con los parámetros predeterminados por la biblioteca.

El modelo Bosques Aleatorios (RF) solo se utilizó en el problema de regresión (*Housing California*), y éste fue reemplazado por GNB en los problemas de clasificación.

En el caso de la Red Neuronal Perceptrón Multicapa (NN) se usó la biblioteca Keras, con TensorFlow (43). La arquitectura de la red consistió en dos capas ocultas de 13 neuronas (este número se eligió mediante prueba y error), utilizando ReLU (44) como la función de activación. Además, se aplicó el algoritmo de optimización Adam (45) para actualizar los pesos de las neuronas durante la fase de entrenamiento, tomando el error cuadrático medio como función de pérdida.

4.2.3. Codificadores evaluados

Para la implementación de los 10 codificadores de atributos categóricos, se empleó la biblioteca *Category Encoders* de Scikit-learn-contrib (46). La Tabla 4.2 muestra las técnicas de codificación seleccionadas para su evaluación, además de su correspondiente abreviatura.

Codificador	**Abreviatura**	**Tipo**	**Aumenta dimensiones**
Ordinal	ORD	No supervisada	No
One hot	ONE	No supervisada	Sí
Sum	SUM	No supervisada	Sí
Helmert	HEL	No supervisada	Sí
Backward Difference	BACK	No supervisada	Sí
Target	TAR	Supervisada	No
M-Estimate	ME	Supervisada	No
Leave one out	LOO	Supervisada	No
CatBoost	CAT	Supervisada	No
James Stein	JST	Supervisada	No

Tabla 4.2: Técnicas de codificación categórica

En esta tabla (4.2), la columna 'Tipo' indica si la técnica usa o no la variable objetivo al mapear el atributo categórico a un número. Además, la columna 'Aumenta dimensiones' indica si para cada instancia en el atributo categórico, se agregará una columna adicional al conjunto de datos de entrenamiento.

4.2.4. Conjuntos de datos

Para evaluar el desempeño de cada combinación modelo-codificador, se emplearon conjuntos de datos reales y sintéticos. A continuación se especifica cómo fueron tratados ambos tipos de conjuntos de datos dentro del marco comparativo.

Conjuntos de datos del mundo real

Los 7 conjuntos de datos reales utilizados para evaluar a los codificadores fueron tomados de la plataforma Kaggle. Estos conjuntos de datos se describen en la Tabla 4.3

Nombre	**Tipo**	**Tuplas**	**Total de atributos**	**Atributos categóricos**	**Cardinalidad**	**Distribución de clases True - False**
Adult	Clas.	48,843	15	9	102	23.9–76.1
Titanic	Clas.	891	12	3	92	38.3–61.4
Credit	Clas.	307,511	122	18	152	91.9–08.1
Employee	Clas.	32,769	10	5	15,626	94.3–05.7
Car auction	Clas.	72,983	34	19	42,370	12.3–87.7
Telco Churn	Clas.	7,043	21	5	4,303	26.5–73.5
Housing California	Reg.	20,610	10	1	5	NA

Tabla 4.3: Descripción de los conjuntos de datos del mundo real

Del total de conjunto de datos, 6 se resolvieron como un problema de clasificación y 1 como un problema de regresión. En la Tabla 4.3, la columna 'Cardinalidad' indica la suma total de instancias encontradas para todos los atributos categóricos en el conjunto de datos. Es importante considerar esta cardinalidad al aplicar aquellas técnicas que pueden añadir columnas adicionales para las instancias de una variable categórica. Por ejemplo, en el caso de los conjuntos de datos 'Employee' y 'Car auction', la técnica de *One Hot Encoding* añadiría 15,626 y 42,370 columnas adicionales respectivamente.

Para todos los conjuntos de datos, se eliminaron los atributos de tipo 'Fecha' y 'IDs'. Además, los valores nulos fueron reemplazados por la media encontrada en el atributo. De igual manera, se aplicó una normalización (47) a los atributos numéricos, es decir, se transformaron para que tuvieran una media de 0 y una desviación estándar de 1, lo que facilita el entrenamiento de los modelos al garantizar que todas las variables estén en la misma escala. Durante el entrenamiento de los modelos, se utilizó la técnica de validación cruzada con una repetición de 3 pliegues (*folds*). Para la generación de los pliegues, se empleó un esquema de validación cruzada estratificada, preservando aproximadamente el mismo porcentaje de muestras de cada clase del atributo objetivo (*target*) en cada conjunto, en comparación con la distribución original del conjunto completo.

La métrica de rendimiento que se empleó fue la exactitud (*Accuracy*) para los problemas de clasificación, y la Raíz del Error Cuadrático Medio (*RMSE*) para el problema

de regresión.

Conjunto de datos sintéticos

Para fines del marco comparativo, el uso de conjuntos de datos del mundo real presenta los siguientes desafíos:

- Contienen imperfecciones que requieren un proceso de limpieza.
- Dado que la función que relaciona los datos con la variable objetivo a predecir es desconocida, es difícil determinar cuál podría ser el mejor rendimiento del modelo.
- Se desconoce de antemano la naturaleza de los atributos categóricos (nominal, ordinal, alta cardinalidad).

Para hacer frente a estos desafíos, se propuso el uso de datos sintéticos. Este tipo de datos han sido utilizados ampliamente en otras áreas de investigación del aprendizaje automático, por ejemplo, para analizar el rendimiento de métodos de selección de características (*Feature Selection*) (48)

En nuestro marco comparativo, se generaron los datos artificialmente a través de funciones predefinidas. Las ventajas de incluir este tipo de datos en nuestro estudio son: 1) Es posible crear un número arbitrario de registros, atributos numéricos y atributos categóricos. 2) Se puede lograr una mejor evaluación del rendimiento de los modelos, ya que se conoce a priori la función que relaciona las variables independientes con la variable dependiente. 3) Se evitan tareas de limpieza de datos. 4) Permite crear problemas de clasificación con diversas propiedades: objetivos (*target*) lineales, no lineales, con clases balanceadas, desequilibradas, etc.

Las funciones utilizadas para generar los datos sintéticos fueron las siguientes (Ecuaciones 4.1,4.2,4.3)

$$f(x, y) = x^2 + 100y \tag{4.1}$$

$$f(x, y) = x^2 + y^2 \tag{4.2}$$

$$f(x, y) = x^3 + y^3 \tag{4.3}$$

Estas funciones fueron seleccionadas de manera arbitraria y como una propuesta inicial. En trabajos futuros, se incluyeron funciones más complejas con atributos categóricos de distintas cardinalidades. Los valores de las variables numéricas x y y se generaron aleatoriamente en el rango [-500, 500]. El número total de tuplas para cada conjunto de datos generado fue de 10,000. Las funciones se evaluaron para cada par (x, y) y el resultado numérico se almacenó en una columna llamada *target*, que fue el valor a predecir por los algoritmos de aprendizaje automático.

Debido a la naturaleza de las funciones, al ser evaluadas para determinar la variable dependiente u objetivo, éstas generan un valor numérico como salida o resultado. Estos valores numéricos fueron procesados para convertir el problema de regresión en uno de clasificación. Para ello, se emplearon dos enfoques:

- Clasificación lineal. Se estableció una única línea de corte tomando el valor promedio en la columna *target*, de tal manera que todos los valores mayores que el promedio se asignaron como Verdadero (o 1), y todos los valores menores que el promedio se asignaron como Falso (o 0);
- 2) Clasificación no lineal. Se establecieron dos líneas de corte de tal manera que los valores numéricos dentro de esas dos líneas se reemplazaron por Verdadero (o 1), mientras que para el resto de instancias que se situaron fuera de esta región, se les asignó el valor de Falso (o 0).

En ambos casos, las líneas de corte se seleccionaron de tal manera que la distribución de clases estuviera balanceada (aproximadamente 50 % Verdadero – 50 % Falso). La Figura 4.1 ilustra los dos enfoques mencionados.

Dado que se busca comprender mejor la efectividad de los codificadores investigados, es necesario mapear los valores numéricos de los atributos 'x' y 'y' a un valor categórico.

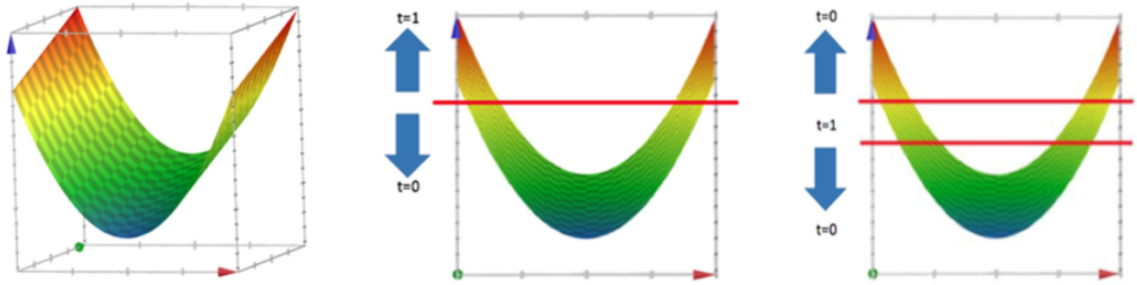

Figura 4.1: *Izquierda*: gráfico de la función $x^2 + 100y$. *Centro*: umbral lineal para el problema de clasificación. *Derecha*: problema de clasificación no lineal.

Esto se realizó añadiendo el sufijo '*_encoded*' a cada valor numérico en la columna a mapear (ver Figura 4.2).

x	y	objetivo
608	966	1
113	421	0
564	362	0
158	860	0
409	776	0
983	167	1

Codificar y

x	y	objetivo
608	966_encoded	1
113	421_encoded	0
564	362_encoded	0
158	860_encoded	0
409	776_encoded	0
983	167_encoded	1

Figura 4.2: Mapeo de un atributo numérico a un atributo categórico.

Los atributos categóricos resultantes son ordinales y con una cardinalidad relativamente alta (10,000 instancias).

El mismo ejercicio se aplicó para la variable x, pero sin codificar y. Esta tarea de mapeo se realizó para las 3 funciones seleccionadas, lo que resultó en 6 conjuntos de datos que fueron incluidos en el estudio comparativo.

Los resultados de esta etapa se encuentran en el Capítulo 5, sección 5.1.

4.3. Etapa II: Propuesta de una nueva técnica de codificación

La metodología descrita en esta sección se encuentra publicada en el artículo 'Cesammo: Categorical encoding by statistical applied multivariable modeling' [(49)].

4.3.1. Fundamentos y justificación

Durante la segunda etapa de la investigación, se propuso analizar la técnica de codificación de CESAMO y con ello diseñar un nuevo codificador basado en la misma. Además, la nueva técnica propuesta fue evaluada y comparada con otras técnicas de codificación mediante el marco comparativo desarrollado en la etapa I.

Se eligió como punto de partida el algoritmo de CESAMO, ya que este tipo de técnicas, basadas en Códigos que Convervan los Patrones (PPCs), han demostrado mantener la relación subyacente en los atributos categóricos. En esencia, CESAMO toma múltiples códigos candidatos de forma aleatoria, estos códigos se evalúan en una función de aproximación (ver Ecuación 3.10), y de esta aproximación se obtiene un error. Aquellos códigos que producen el menor error se consideran los PPCs.

Es importante resaltar que CESAMO se basa en un muestreo de doble nivel, es decir, al realizar la aproximación con la función polinómica, solo se están considerando dos variables: 1) el atributo categórico que se desea codificar, y 2) otra variable elegida al azar en el conjunto de datos. Esta segunda variable, seleccionada al azar en cada iteración del algoritmo, permite explorar el espacio de todas las variables del conjunto de datos.

4.3.2. Diseño del codificador propuesto

Dado que CESAMO realiza una aproximación de 2 variables, en última instancia solo se preserva la relación entre el atributo categórico y la segunda variable con la que la función de aproximación obtuvo el menor error. En otras palabras, de existir

otra u otras relaciones entre el atributo categórico y las demás variables (más allá de la encontrada por CESAMO), estas no se reflejarían en los PPCs.

Para evitar esta posible pérdida de información, se propuso CESAMMO: *Categorical Encoding by Statistical Applied Multivariable Modeling*. Nuestra hipótesis inicial fue que introduciendo un factor multivariable se mejoraría notablemente el desempeño del algoritmo en comparación con la versión original. Este factor multivariable se basa en considerar la relación entre el atributo categórico y todas las demás variables, mediante el cálculo de los promedios de los errores cuando un conjunto de códigos candidatos se valúan con todos los atributos categóricos.

En la Figura 4.3 se muestra el diagrama de flujo de CESAMMO.

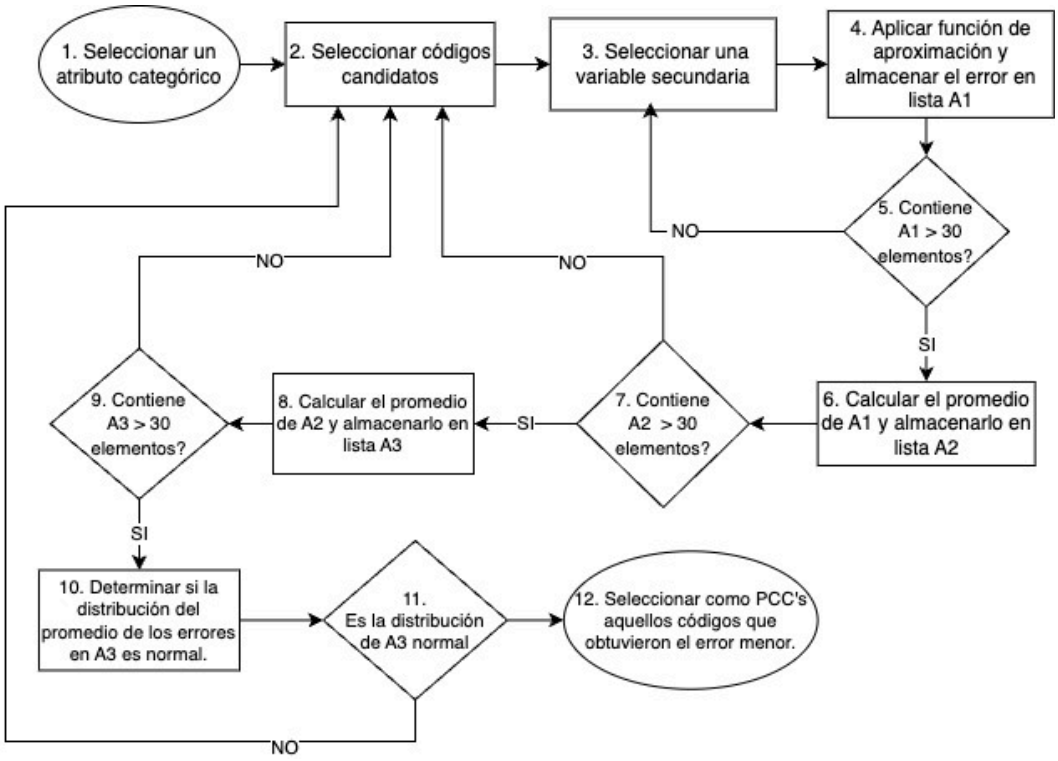

Figura 4.3: Diagrama de flujo del algoritmo de CESAMMO

El algoritmo de CESAMMO puede resumirse de la siguiente manera:

1. Seleccionar un atributo categórico.
2. Definir códigos candidatos al azar para reemplazar temporalmente las instancias categóricas.

3. Seleccionar aleatoriamente una variable secundaria en el conjunto de datos.

4. Aplicar la función de aproximación polinómica entre la variable categórica y la variable secundaria seleccionada. Almacenar el error resultante en una lista (A1).

5. Verificar si el tamaño de A1 es mayor que 30. Si es verdadero, proceder al paso 6. De lo contrario, repetir el paso 3.

6. Calcular el promedio de A1 y almacenar el resultado en una nueva lista (A2).

7. Verifica si el tamaño de A2 es mayor que 30. Si es verdadero, proceder al paso 8. De lo contrario, repetir el paso 2.

8. Calcular el promedio del A2 y almacenar el resultado en una nueva lista (A3).

9. Verificar si el tamaño de la lista A3 es mayor que 30. Si es verdadero, proceder al paso 10. De lo contrario, repetir el paso 2.

10. Determinar (calcular), a través de la prueba Chi-cuadrada (50), si la distribución de los promedios de los errores en A3 es normal.

11. Si la distribución de A3 es normal, proceder al paso 12, de lo contrario, repetir el paso 2.

12. Seleccionar los PPCs, los cuales son aquellos códigos candidatos que obtuvieron el menor **error promedio** al ser evaluados con todos los atributos del conjunto de datos.

La modificación principal que se realizó al algoritmo de CESAMO se encuentra en los pasos 5, 6 y 7. Se asumió que al tomar el **promedio de los errores** entre el atributo categórico y todas las demás variables en el conjunto de datos de entrenamiento, los PPCs mantendrían el patrón subyacente de todo el conjunto de datos. Esto debería, en principio, mejorar el rendimiento de los modelos de aprendizaje automático.

4.3.3. Evaluación del codificador propuesto

El siguiente paso en la metodología fue cuantificar y comparar el rendimiento de CESAMMO vs CESAMO (versión original). Para ello se utilizó el marco comparativo desarrollado en la Etapa I 4.2. Se incluyeron en el marco comparativo un total de 12 codificadores categóricos y 5 algoritmos de aprendizaje automático. Además de CESAMO y su versión multivariable, las codificadores comparados fueron los siguientes:

- *Ordinal Encoder*
- *One-Hot Encoder*
- *Sum Encoder*
- *Helmert Encoder*
- *Backward Difference Encoder*
- *Target Encoder*
- *M-Estimate Encoder*
- *Leave One Out Encoder*
- *James Stein Encoder*
- *CatBoost Encoder*

Por otro lado, los algoritmos de aprendizaje automático incluidos fueron:

- Regresión Logística
- Gaussian Naïve-Bayes
- Máquina de Soporte Vectorial
- Red Neuronal Perceptrón Multicapa
- XGBoost

Los hallazgos obtenidos en esta etapa se presentan en el Capítulo 5, Sección 5.2. Dado que los resultados de esta etapa influyeron en la siguiente, es relevante adelantar que, en resumen, no se encontró una diferencia significativa en el desempeño de CESAMO Y CESAMMO, ya que ambos obtuvieron resultados muy similares. Además, al tener que tomar el promedio de los errores de los códigos candidatos, esto implicó que la versión multivariable fuese más costosa en términos computacionales. Este costo computacional fue la razón por la que se descartó a CESAMMO como objeto de estudio en la Etapa III, y la investigación se centrase únicamente en CESAMO.

4.4. Etapa III: Evaluación estadística de codificadores

La metodología descrita en esta sección se encuentra publicada en el artículo 'Statistical Evaluation of Categorical Encoders for Pattern Preservation in Machine Learning Tasks.' [(51)].

En esta tercera y última etapa, la investigación se centró en llevar a cabo un análisis más formal de las técnicas de codificación, incluida CESAMO. Para ello, se planteó una evaluación respaldada por un criterio estadístico. A continuación, se detalla la metodología utilizada.

4.4.1. Justificación

Existen en la literatura múltiples estudios que han medido y comparado el rendimiento de las diversas técnicas de codificación. Los más relevantes hasta la fecha se expusieron en el Capítulo 3 (3.5). Estos estudios ofrecen evidencia que indica que el rendimiento de algunos codificadores es superior que el de otros. Sin embargo, todos estos estudios tienen la limitante de que los resultados no están respaldados por una base estadística. Es decir, en estos estudios, los codificadores son evaluados eligiendo de manera subjetiva un número predeterminado de conjuntos de datos, siendo la evaluación más amplia hasta la fecha (2024) el trabajo (38), el cual incluye 50 conjuntos de datos.

Para obtener conclusiones más robustas, los codificadores deben de evaluarse con tantos conjuntos de datos como sean necesarios hasta que se cumpla cierto criterio que esté justificado estadísticamente.

Con base en la desventaja previamente mencionada, surgió la necesidad de diseñar una metodología que permitiera evaluar con rigor estadístico las técnicas de codificación.

4.4.2. Fundamentos de la metodología

Para evaluar y comparar a las técnicas de codificación, se propuso emplear como métrica el **Error Mínimo (*EM*)**, la cual es una medida estadística que cuantifica el rendimiento de cualquier método. En las siguientes líneas se explican sus fundamentos.

Cada codificador está asociado a una **Distribución de Errores (*DE*)**, la cual representa el conjunto de todos los posibles errores que pueden surgir al aplicar la técnica a distintos problemas. Esta *DE* se caracteriza por su media μ_{DE} y su desviación estándar σ_{DE}. La métrica del Error Mínimo se define como el valor posicionado a k desviaciones estándar a la izquierda de la media dentro de la Distribución de Errores (Ecuación 4.4)

$$EM = \mu_{DE} - k\sigma_{DE} \tag{4.4}$$

Dado que el Error Mínimo depende de conocer de antemano μ_{DE} y σ_{DE}, es necesario entonces determinar primero la Distribución de Errores de la técnica de codificación a evaluar. Para lograr esto, se evaluaron los codificadores en tantos conjuntos de datos hasta que se cumplió el criterio estadístico. En esencia, este criterio estadístico se basa en obtener errores al evaluar el codificador hasta que la distribución del promedio de estos errores se convierta en una distribución normal. Se sabe por el Teorema del Límite Central que esto siempre ocurrirá ([52]). En la siguiente sección se describen los principios de la métrica Error Mínimo elegida en esta metodología

4.4.3. Métrica Error Mínimo

Esta métrica se fundamenta en la desigualdad de Chebyshev, que establece que, para cualquier conjunto de datos, al menos una proporción de las muestras se encuentra a una distancia menor o igual a k veces la desviación estándar de la media, independientemente de la forma de la distribución (ya sea normal, bimodal, uniforme o cualquier otra) (53). En la Ecuación 4.5 se muestra la desigualdad, la cual se puede interpretar como: sea X una variable aleatoria con media μ y desviación estándar σ, la probabilidad de que una observación se desvíe de la media más de $k\sigma$ es, como máximo, $\frac{1}{k^2}$.

$$P\left(|X-\mu| \geq k\sigma\right) \leq \frac{1}{k^2} \tag{4.5}$$

Seleccionando $k = \sqrt{5}$, la desigualdad de Chebyshev establece que la probabilidad de que X se desvíe de su media en más de $\sqrt{5}\sigma$ es, como máximo, del 20 %. Esto se ilustra en la Figura 4.4, donde la línea vertical punteada roja denota el valor situado en $-\sqrt{5}\sigma$, y la línea verde el valor en $\sqrt{5}\sigma$. Las áreas sombreadas en las colas de la distribución representan este límite superior del 20 % de los datos. De esta manera, la desigualdad garantiza que al menos el 80 % de los datos se encuentra dentro de estas dos líneas verticales punteadas.

Supongamos que la distribución mostrada en la Figura 4.4 representa la distribución de errores de una técnica de codificación. Como el objetivo inicial era encontrar una métrica estadística para comparar a los codificadores, nuestro enfoque se centró en determinar el error situado a $-k$ desviaciones estándar de la media (cuanto menor es el error, mejor es el desempeño la técnica de codificación). Por lo tanto, se definió el Error Mínimo como el valor ubicado en $\mu - \sqrt{5}\sigma$, lo que corresponde a la línea vertical punteada roja en la Figura 4.4.

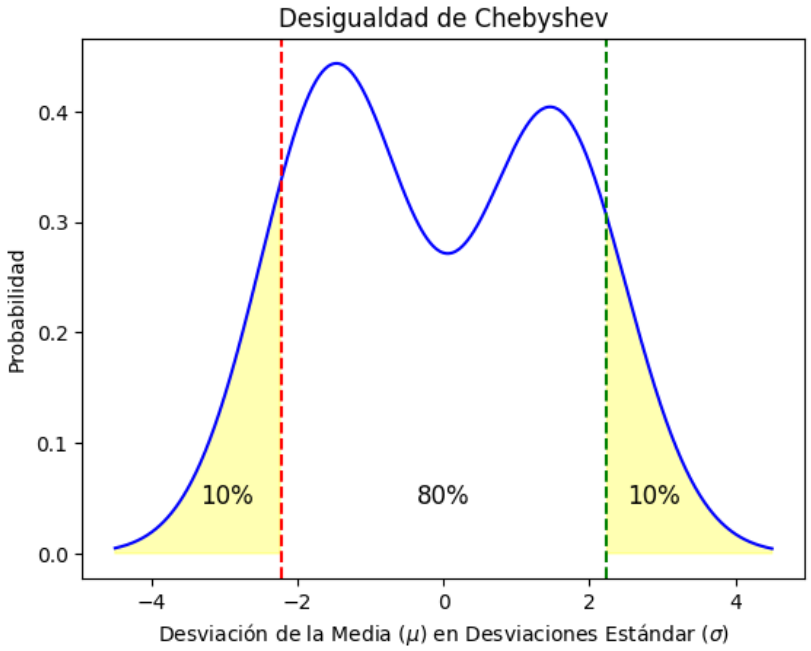

Figura 4.4: Desigualdad de Chebyshev seleccionando $k = \sqrt{5}$

Para nuestra evaluación, esto implica reemplazar k por $\sqrt{5}$ en la Ecuación 4.4

4.4.4. Generación de datos sintéticos

Para determinar la Distribución de Errores (DE) de una técnica de codificación, se requiere evaluarla con tantos conjuntos de datos hasta que la distribución del promedio de los errores obtenidos se convierta en una distribución normal. Lograr esto con conjuntos de datos del mundo real representa un desafío, ya que se desconoce a priori cuantos conjuntos de datos se requieren para alcanzar este criterio estadístico, y, por otro lado, el número de conjuntos de datos del mundo real accesibles es limitado. Para sortear este desafío, empleamos conjuntos de datos sintéticos.

Los conjuntos de datos sintéticos fueron generados mediante funciones polinómicas. Estos datos contienen tanto atributos numéricos como categóricos. El código para generar los datos sintéticos se encuentra disponible en: `https://github.com/celestun/polynomial-dataset-generator`).

Para emular las propiedades normalmente encontradas en conjuntos de datos del mundo real, se analizaron 10 conjuntos representativos y a partir de esto, se definieron

ciertas propiedades, como el número total de atributos numéricos y categóricos, tamaño del conjunto de datos, complejidad del objetivo (lineal, no lineal), entre otros. Los conjuntos de datos reales que se analizaron se encuentran en el apartado de Apéndice, sección 2.

El propósito de estas propiedades es definir los parámetros de los polinomios que generaron los datos sintético. Estos parámetros se muestran en la Tabla 4.4.

Nombre del parámetro	**Rango**	**Descripción**
Número de variables: x	$\{x \mid v \in N, 5 \leq v \leq 34\}$	Número de atributos en el conjunto de datos sintético.
Dominio de las variables: d	$\{d \mid d \in R, 0 \leq d \leq 1\}$	El dominio de los atributos.
Grados del polinomio: P_d	$\{P_d \mid P_d \in N, 0 \leq P_d \leq 11\}$	El exponente de cada variable en el polinomio.
Términos del polinomio: P_t	$\{P_t \mid P_t \in N, 1 \leq P_t \leq 13\}$	El número de términos del polinomio.
Valores de los coeficientes: c	$\{c \mid c \in R, -1 \leq c \leq 1\}$	El valor que multiplica cada término.
Número de variables en cada término: n_x	$\{n_x \mid n_x \in N, 1 \leq n_x \leq 5\}$	El número de variables en cada término.
Tamaño del conjunto de datos: n	$\{n \mid n \in N, 1000 \leq n \leq 3000\}$	El número de tuplas.

Tabla 4.4: Parámetros polinomiales para generar datos sintéticos

Durante la implementación del generador de datos sintéticos, los parámetros polinomiales se asignaron de manera aleatoria dentro de rangos predefinidos (ver columna 'Rango' en la Tabla 4.4). Esto asegura que cada conjunto de datos generado sea único. Dado que el propósito de generar datos sintéticos era evaluar los codificadores categóricos, se introdujeron atributos categóricos en los datos. Del análisis previo realizado a los conjuntos de datos del mundo real, se observó que aproximadamente el 30 % de las variables son categóricas.

4.4.5. Evaluación de los codificadores

Para la evaluación de los codificadores, se amplió el marco comparativo elaborado en la Etapa I 4.2. Éste se modificó para incorporar el criterio de evaluación estadístico.

Los datos sintéticos se incluyeron al marco comparativo como un problema de clasificación binaria, manteniendo una distribución de clases balanceada (50 %-50 %) en la variable a predecir.

Dado que los conjuntos de datos estaban balanceados, se utilizó la métrica de exactitud (acc) para medir el rendimiento de los modelos al resolver los problemas sintéticos, y luego se definió el error (e) como $e = 1 - acc$.

Respecto a los modelos de aprendizaje automático, se incluyeron los siguientes: una Red Neuronal Perceptrón Multicapa, una Regresión Logística, una Máquina de Soporte Vectorial, el modelo Gaussian Naive-Bayes y XGBoost.

Por otro lado, los codificadores evaluados fueron: CESAMO, *Binary, Hashing, One-hot, Ordinal* y *Count encoders*. El código del marco comparativo de esta etapa se encuentra disponible en `https://github.com/celestun/categorical-encoders-benchmark`.

Cada modelo de aprendizaje automático resolvió los problemas empleando todos los distintos codificadores, por lo que se obtuvieron 30 combinaciones distintas de modelo-codificador.

Para cada combinación de modelo-codificador, se siguieron los siguientes pasos generales para obtener las métricas de rendimiento:

1. Crear un conjunto de datos sintético usando la función polinómica.
2. Aplicar la técnica de codificación y reemplazar las instancias categóricas en el conjunto de datos.
3. Resolver el conjunto de datos codificado con el modelo y registrar el error resultante.
4. Repetir los pasos 1, 2 y 3 hasta alcanzar 36 errores.
5. Calcular y registrar la media de los errores.

6. Verificar si la distribución de la media de los errores es normal. Si esto se cumple, proceder al paso 7; de lo contrario, regresar al paso 1.

7. Concluir el proceso de muestreo y determinar la media y la desviación estándar de la distribución de la media de los errores.

En el paso 4, el tamaño de una muestra de 36 elementos parte de una regla general basada en el Teorema del Límite Central, que establece que, a medida que aumenta el tamaño de la muestra, la distribución de las muestras tiende a aproximarse a una distribución normal, independientemente de la distribución subyacente de la población (54). En el paso 6, para verificar si la distribución era normal, se implementó la Distribución Л, la cuál se describe a continuación.

4.4.6. Distribución Л

Un paso fundamental en la evaluación estadística de las técnicas de codificación es comprobar si la distribución del promedio de los errores es una distribución normal.

Existen en la literatura varios métodos comúnmente utilizados para evaluar la normalidad de una distribución. Estos métodos, llamados pruebas de bondad de ajuste (*GoF* por sus siglas en inglés) son herramientas estadísticas que se utilizan para determinar si los datos observados en una muestra son consistentes con una distribución teórica esperada, como una distribución normal (en nuestro caso). Las pruebas de bondad de ajuste más populares son Chi-cuadrada (χ^2) (50), Shapiro-Wilk (55), Anderson-Darling (56) y Kolmogorov-Smirnov (57). Si bien estas pruebas son ampliamente usadas para corroborar normalidad, es importante resaltar una limitante que presentan.

Para explicar la limitante, consideremos que en estas pruebas, la hipótesis nula (H_0) es que los datos siguen una distribución normal. Estas pruebas evalúan si hay suficiente evidencia para rechazar o no la hipótesis nula. Si el valor p es menor que un umbral (normalmente, 0.05), se rechaza la hipótesis nula, lo que indica que los datos no se ajustan a una distribución normal.

El problema radica en que no rechazar la hipótesis nula no significa que la distribución sea confirmada como normal. Solo significa que no se encontró suficiente evidencia

para rechazar la hipótesis con base en los datos observados. Esto es un punto crucial en estas pruebas estadísticas. No rechazar H_0 no implica que H_0 sea verdadera (es decir, no implica que los datos estén normalmente distribuidos), sino que no hay suficiente evidencia para concluir lo contrario. En otras palabras, la ausencia de evidencia para rechazar la normalidad no confirma normalidad, más bien solo indica que el conjunto de datos podría ser normal, pero no hay certeza. En nuestro marco comparativo, el objetivo es confirmar si una distribución es normal, por lo que se necesita una alternativa a estas pruebas de bondad de ajuste. Con el fin de resolver la limitante mencionada anteriormente, se propuso implementar la distribución Λ (58). Esta distribución asegura, con una cierta probabilidad, que una muestra proviene de una distribución normal. El fundamento de la distribución Λ se basa en la pregunta: ¿Qué tan probable es calcular un valor experimental de Λ mayor que ξ para una muestra que se espera sea normal?, donde Λ es el valor obtenido después de evaluar los datos bajo la Ecuación 4.6 , y ξ es un valor crítico predefinido. Para responder a esta pregunta, en el caso de nuestra metodología, la distribución del promedio de los errores se divide en cuantiles. Cada cuantil abarca la misma área bajo la curva de distribución. La Figura 4.5 proporciona un ejemplo donde la distribución se segmenta en deciles (10 cuantiles).

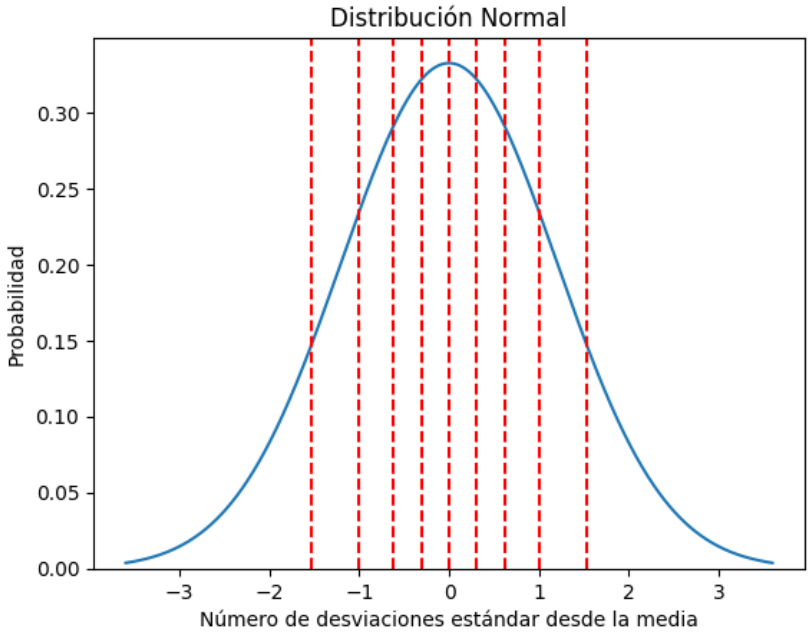

Figura 4.5: Distribución normal segmentada en deciles. Cada decil abarca el 10 % del área bajo la curva.

La Ecuación 4.6 expresa cómo se pueden obtener valores de $\beth$.

$$\beth = \sum_{i=1}^{Q} \frac{(O_i - E_i)^2}{E_i} \wedge [O_i \geq \Phi \ \forall i] \tag{4.6}$$

Donde

- Q el número de cuantiles.
- O_i denota el número de eventos observados en el cuantil i-ésimo.
- E_i es el número de observaciones esperadas en el cuantil i-ésimo
- Φ es el número mínimo de observaciones requeridas por cuantil

Una vez que se tiene un valor especifico de $\beth$, cuanto menor sea este valor, más cercanas estarán las muestras a los valores esperados para que la distribución se pueda considerar una distribución normal. Por lo tanto, si $\beth \leq \xi$, entonces los datos están distribuidos normalmente con una probabilidad $> 1 - p$.

Los resultados de esta evaluación estadística que se plantea en esta Etapa III se exponen en el Capítulo 5, Sección 5.3

Capítulo 5

Resultados experimentales

En este capítulo se presentan y analizan los resultados obtenidos durante el transcurso del proyecto de doctorado. Como se mencionó en el capítulo anterior (4), la investigación se compuso de 3 etapas principales, cuyos objetivos fueron:

- Etapa I: Estudio y cuantificación del efecto de las distintas técnicas de codificación en los modelos de aprendizaje automático.
- Etapa II: Diseño y desarrollo de una propuesta de un codificador basado en CESAMO.
- Etapa III: Implementación de una evaluación respaldada estadísticamente de las técnicas de codificación.

En las siguientes secciones se presentan los resultados experimentales obtenidos para cada etapa.

5.1. Resultados Etapa I: Desarrollo de marco comparativo

Los resultados expuestos en esta sección se encuentran publicados en el artículo 'Measuring the Effect of Categorical Encoders in Machine Learning Tasks Using Synthetic Data' [(42)] .

Resumen metodología: de manera breve, en esta etapa se implementó un marco comparativo diseñado para evaluar el impacto de 10 diferentes técnicas de codificación de atributos categóricos en el rendimiento de 5 modelos de aprendizaje automático. El estudio se realizó utilizando tanto conjuntos de datos del mundo real como conjuntos de datos generados de forma sintética. Se entrenaron cinco algoritmos de aprendizaje automático: Regresión Logística, Gaussian Naïve Bayes, Máquinas de Soporte Vectorial, Redes Neuronales y XGBoost, donde cada modelo se combinó con las 10 diferentes técnicas de codificación. Los resultados obtenidos del marco comparativo se exponen en las siguientes secciones.

5.1.1. Conjuntos de datos del mundo real

En las Figuras 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 se ilustran los resultados obtenidos de los conjuntos de datos del mundo real implementados en el marco comparativo. Cada figura contiene 3 gráficas:

- La gráfica de la izquierda en color verde muestra las combinaciones modelo-codificador que tuvieron el rendimiento más alto.
- La gráfica central en color rojo indica las combinaciones modelo-codificador con el rendimiento más bajo.
- La gráfica de la derecha en color azul muestra la diferencia (Δ) entre el mejor y el peor rendimiento obtenido por cada modelo.

La métrica más relevante, Δ, cuantifica en qué medida las técnicas de codificación pueden afectar al mismo algoritmo de aprendizaje automático.

Para el caso de los conjuntos de datos de *Employee* (Figura 5.3) y *Car Auction* (Figura 5.4), fue necesario excluir a las técnicas que aumentan el tamaño del conjunto de datos (como *One-hot encoding*). Esto debido a la alta cardinalidad en los atributos categóricos presentes en ambos conjuntos de datos.

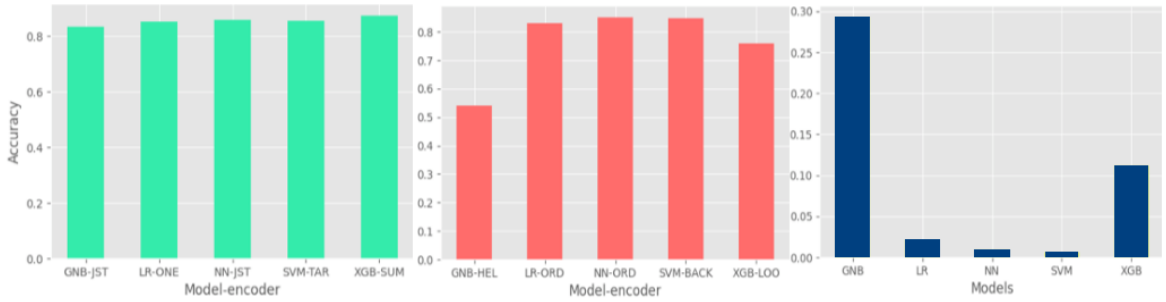

Figura 5.1: Resultados para el conjunto de datos *Adult*

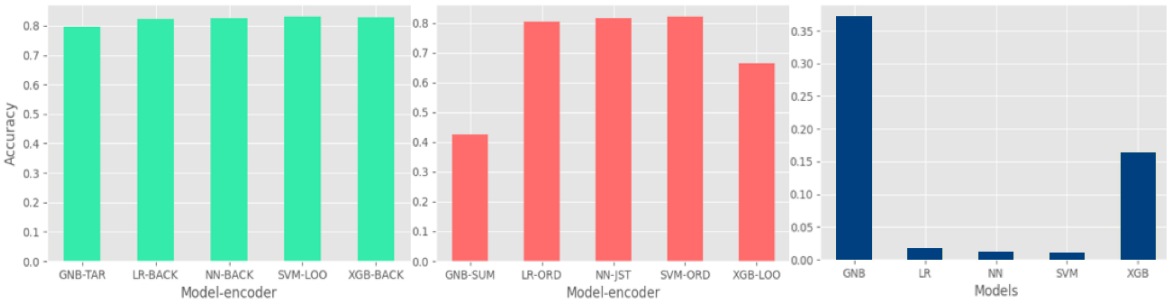

Figura 5.2: Resultados para el conjunto de datos *Titanic*

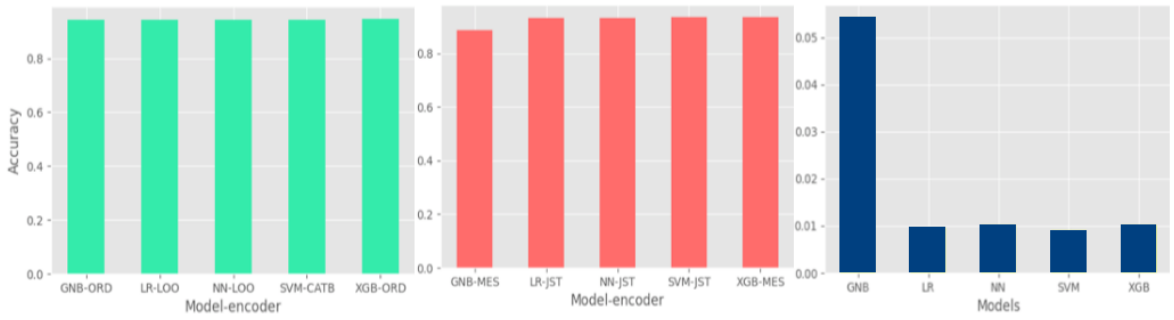

Figura 5.3: Resultados para el conjunto de datos *Employee*

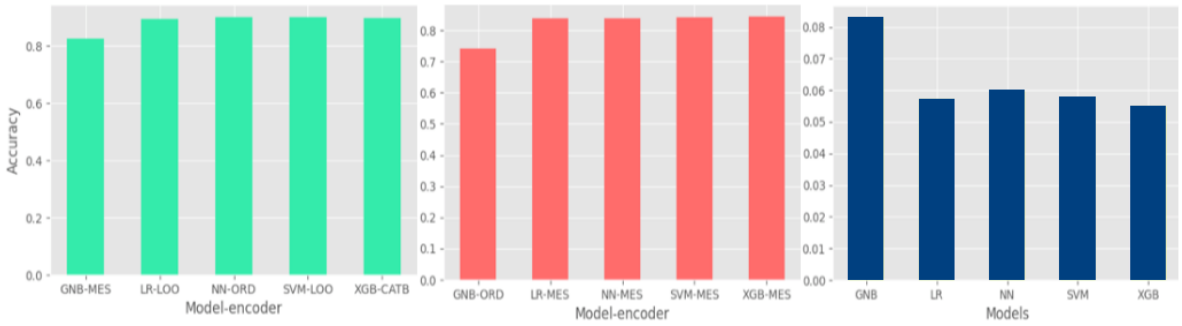

Figura 5.4: Resultados para el conjunto de datos *Car Auction*

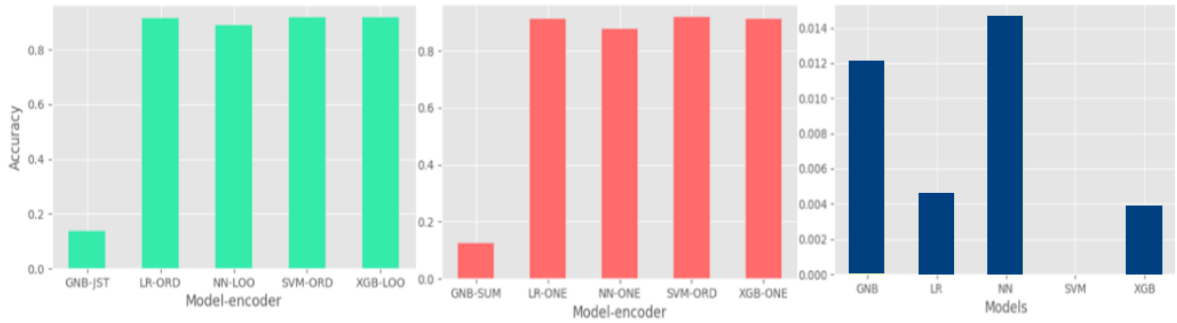

Figura 5.5: Resultados para el conjunto de datos *Credit data*

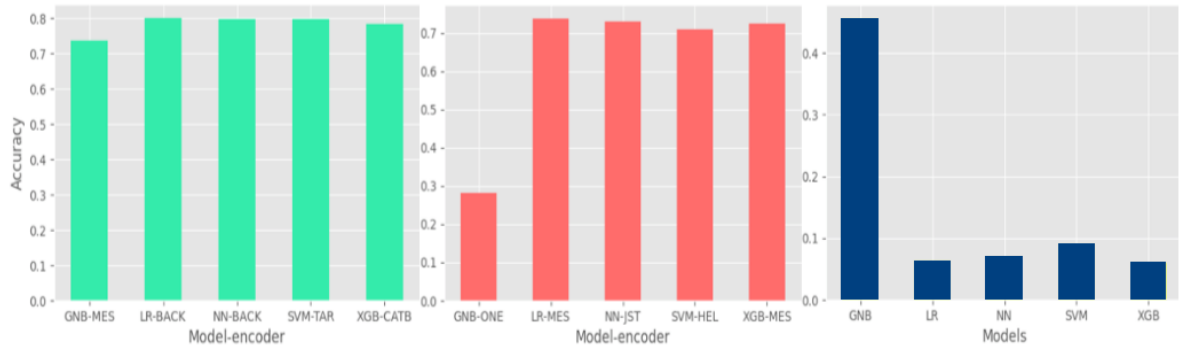

Figura 5.6: Resultados para el conjunto de datos *Telco Churn*

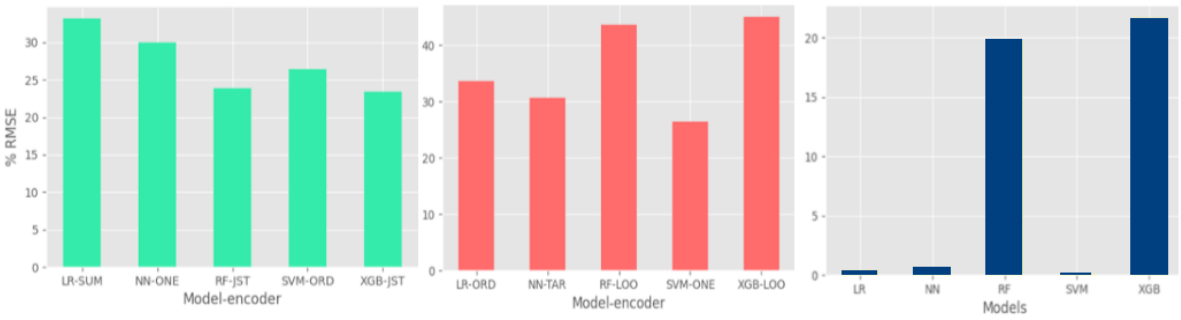

Figura 5.7: Resultados para el conjunto de datos *Housing California*

De los resultados obtenidos, se destaca lo siguiente. Uno de los comportamientos que ocurrieron con mayor frecuencia fue que la Red Neuronal Perceptrón Multicapa resultó ser el modelo menos afectados por los diferentes codificadores. De manera similar a las Redes Neuronales, las Máquinas de Soporte Vectorial y la Regresión Logística se vieron ligeramente afectados, independientemente de la técnica de codificación.

En contraste, el modelo Gaussian Naive-Bayes fue el que se vio más afectado de manera negativa, particularmente por aquellos métodos que añaden columnas para cada instancia.

También se encontró que las técnicas de codificación que obtuvieron con mayor frecuencia los peores rendimientos fueron

- *Ordinal Encoder*: Figuras 5.1 , 5.2, 5.4, 5.5, 5.7
- *M-Estimate encoder*: Figuras 5.3, 5.4, 5.6

5.1.2. Conjuntos de datos sintéticos

En esta sección se muestran los resultados obtenidos a partir de la evaluación con datos sintéticos generados con las funciones 4.1, 4.2, 4.3.

Es relevante resaltar que estas funciones, al ser evaluadas, dan como resultado valores numéricos en la variable objetivo. Por otro lado, como el propósito de estos datos es cuantificar el desempeño de los codificadores, se reemplazaron los atributos numéricos (x o y) por atributos categóricos. Finalmente, la variable objetivo se reemplazó por valores de '0' o '1' con base en un criterio lineal o no lineal.

En resumen, cada ecuación produjo 6 conjuntos de datos con distintas características y propósitos (configuraciones). A continuación se describe cada uno de los conjuntos de datos sintéticos generados para cada ecuación.

1. Conjunto de datos con únicamente atributos numéricos. La variable objetivo se generó con un criterio lineal. El propósito de este conjunto de datos, al no tener atributos categóricos, es generar valores de referencia (su uso se explica más adelante).

2. Conjunto de datos con únicamente atributos numéricos. La variable objetivo se generó con un criterio NO lineal. Propósito: generar valores de referencia.

3. Conjunto de datos con la variable x mapeada a valores categóricos. La variable y se mantuvo numérica. La variable objetivo se generó de manera lineal. Propósito: evaluar a los codificadores.

4. Conjunto de datos con la variable y mapeada a valores categóricos. La variable x se mantuvo numérica. La variable objetivo se generó de manera lineal. Propósito: evaluar a los codificadores.

5. Conjunto de datos con la variable x mapeada a valores categóricos. La variable y se mantuvo numérica. La variable objetivo se generó de manera NO lineal. Propósito: evaluar a los codificadores.

6. Conjunto de datos con la variable y mapeada a valores categóricos. La variable x se mantuvo numérica. La variable objetivo se generó de manera NO lineal. Propósito: evaluar a los codificadores.

Para cada configuración, se aplicaron las 50 combinaciones diferentes de modelo-codificador. En cuanto a los conjuntos de datos provenientes de las configuraciones (1) y (2) del listado anterior, se utilizaron para generar valores de referencia. Es decir, al ser aplicados directamente (sin atributos categóricos) a los modelos, estos nos permiten identificar el desempeño más alto (relativo) que un modelo puede alcanzar al tener acceso a la información sin codificar. Pues se espera que, después de mapear los valores

numéricos a una categoría (añadiendo el prefijo '_encoded'), los desempeños obtenidos sean iguales o menores que los valores de referencia.

Como parte de los resultados se observó que, en el caso de la función $x^2 + 100y$, al resolver la configuración 1), todos los modelos de aprendizaje automático mostraron una exactitud casi perfecta al predecir la variable objetivo de comportamiento lineal.

De manera similar, con la configuración 2) la mayoría de los modelos lograron alrededor de un 90 % de exactitud para el objetivo no lineal (exceptuando la Regresión Logística). La Tabla 5.1 muestra los resultados obtenidos para los valores de referencia.

Tabla 5.1: Resultados para la función x^2+100y con solo valores numéricos (valores de referencia)

Algoritmos de aprendizaje automático	**Exactitud para objetivo lineal (%)**	**Exactitud para objetivo no lineal (%)**
XGBoost (XGB)	100	100
Red Neuronal (NN)	99.91	99.37
Gaussian Naïve-Bayes (GNB)	99.80	89.16
Regresión Logística (LR)	99.79	61.51
Máquina de Soporte Vectorial (SVC)	99.74	98.85

En las Tablas 5.2 y 5.3 se muestran los resultados para las configuraciones 3) y 4). Al igual que en los resultados de los conjuntos de datos del mundo real, para cada modelo de aprendizaje automático, se registró el codificador que obtuvo el mejor y el peor rendimiento, así como la diferencia entre estos dos valores. Cuando un modelo obtuvo el mismo rendimiento independientemente del codificador, se usó el signo '—', lo que significa que el modelo no fue afectado en absoluto por la técnica de codificación.

Tabla 5.2: Resultados para $x^2 + 100y$ con x mapeado a valores categóricos y con objetivo lineal

Modelo	Cod. con mejor rendimiento	Mejor exactitud	Cod. con peor rendimiento	Peor exactitud	Dif.
NN	ORD	100	HEL	95.29	4.71
XGB	—	100	—	100	0
LR	ORD	99.79	HEL	95.39	4.4
SVC	ORD	99.64	HEL	89.49	10.15
GNB	ORD	98.34	BACK	56.5	41.84

Tabla 5.3: Resultados para $x^2 + 100y$ con y mapeado a valores categóricos y con objetivo lineal

Modelo	Cod. con mejor rendimiento	Mejor exactitud	Cod. con peor rendimiento	Peor exactitud	Dif.
XGB	—	100	—	100	0
NN	CAT	99.98	SUM	93.82	6.15
LR	CAT	99.81	HEL	95.96	3.85
SVC	CAT	99.57	ONE	75.57	24
GNB	CAT	98.84	SUM	45.57	53.26

De la Tabla 5.2, se puede observar que la técnica de *Ordinal Encoding* obtuvo el mejor rendimiento. Esto podría deberse a que la variable x, antes de ser mapeada a un atributo categórico, tenía un orden numérico. Por lo tanto, probablemente los valores asignados por *Ordinal Encoding* coincidieron con el orden original de la variable x.

De manera similar, las Tablas 5.4 y 5.5 muestran los resultados para las configuraciones 6) y 7) (objetivo no lineal).

Tabla 5.4: Resultados para $x^2 + 100y$ con x mapeado a valores categóricos y con objetivo no lineal

Modelo	Cod. con mejor rendimiento	Mejor exactitud	Cod. con peor rendimiento	Peor exactitud	Dif.
XGB	ORD	100	JST	99.98	0.02
NN	ME	99.39	ONE	88.55	10.84
SVC	ORD	98.61	BACK	63.34	35.28
GNB	ORD	91.15	BACK	53.06	38.09
LR	HEL	79.34	ORD	61.56	17.78

Tabla 5.5: Resultados para $x^2 + 100y$ con y mapeado a valores categóricos y con objetivo no lineal

Modelo	**Cod. con mejor rendimiento**	**Mejor exactitud**	**Cod. con peor rendimiento**	**Peor exactitud**	**Dif.**
XGB	—	100	—	100	0
NN	CAT	99.59	ONE	80.87	18.71
SVC	ORD	98.71	BACK	53.65	45.06
GNB	CAT	89.15	SUM	53.12	36.03
LR	ONE	59.17	CAT	56.55	2.63

Las mediciones obtenidas para las funciones $f(x, y) = x^2 + y^2$ y $f(x, y) = x^3 + y^3$ se muestran en la sección de Apéndice 1

De los resultados obtenidos, se puede observar que los modelos de Máquinas de Soporte Vectorial (SVM), Gaussian Naive-Bayes (GNB) y la Red Neuronal (NN) se vieron negativamente afectados por aquellos codificadores que aumentan el tamaño del conjunto de datos.

En contraste, se encontró que el modelo de Regresión Logística, para las configuraciones con un objetivo no lineal, funcionó mejor con aquellos codificadores que incrementan el tamaño de los datos, e incluso en algunos casos superaron los valores de referencia.

Las Tablas 5.6 y 5.7 muestran a los codificadores que obtuvieron con mayor frecuencia los mejores y peores rendimientos en las diferentes combinaciones.

Tabla 5.6: Codificadores evaluados con datos sintéticos con los mejores rendimientos

Técnica de codificación	**Total de primeros lugares (de 55)**	**% de obtención de 1ros lugares**
CatBoost	15	27.27
Leave one out	14	25.45
Ordinal	11	20.00
One hot encoder	5	9.09
Helmert	4	7.27
Sum	2	3.63
M estimate	2	3.63
James Stein	1	1.81
Backward difference	1	1.81

Tabla 5.7: Codificadores evaluados con datos sintéticos con los peores rendimientos

Técnica de codificación	**Total de últimos lugares (de 55)**	**% de obtención de últimos lugares**
Backward difference	22	40.00
Ordinal	13	23.63
Sum	8	14.54
Helmert	5	9.09
One hot encoder	4	7.27
CatBoost	2	3.63
James Stein	1	1.81

Como se puede observar en las tablas, la técnica de codificación *Catboost* se encuentra en el primer lugar en la lista de los mejores rendimientos. Por otro lado, la técnica *Backward difference* se encuentra entre los codificadores con los desempeños más bajos.

5.2. Resultados Etapa II: Propuesta de una nueva técnica de codificación

Los resultados expuestos en esta sección se encuentran publicados en el artículo 'CESAMMO: Categorical Encoding by Statistical Applied Multivariable Modeling' [(49)].

Resumen metodología: de manera concisa, en esta etapa se buscó evaluar y comparar a la técnica de codificación de CESAMMO: *Categorical Encoding by Statistical Applied Multivariable Modeling*. A diferencia de la versión original de CESAMO, esta nueva versión propuesta se basa en determinar la relación entre el atributo categórico y todas las demás variables del conjunto de datos.

Para la evaluación de la técnica propuesta (CESAMMO), se resolvieron 6 problemas de clasificación con 5 modelos de aprendizaje automático distintos. Para cada modelo, se implementaron 12 técnicas de codificación diferentes (incluyendo CESAMO y CESAMMO), y se midió la exactitud de cada combinación modelo-codificador. La exactitud se calculó dividiendo el número de predicciones correctas entre el número total de predicciones. Los resultados obtenidos se presentan a continuación.

Las Tablas 5.8, 5.9, 5.10, 5.11, 5.12 y 5.13 muestran los hallazgos obtenidos para cada conjunto de datos empleado en la evaluación. Las columnas en cada tabla describen lo siguiente:

- **Modelo**: El algoritmo de aprendizaje automático entrenado con el cual se resolvió el problema.
- **Exactitud CESAMO**: La exactitud obtenida por CESAMO.
- **Exactitud CESAMMO**: La exactitud obtenida por CESAMMO.
- **C1**: Contiene el nombre de la técnica de codificación que obtuvo la mejor exactitud.
- **Exactitud C1**: El valor numérico de la mejor exactitud obtenida por el mejor codificador.
- **Δ(CMMO, C1)**: Expresa la diferencia absoluta entre CESAMMO y la técnica de codificación que obtuvo la mejor exactitud.

Para interpretar de manera más clara los resultados, la primera fila de la Tabla 5.8 podría leerse como:

"Al resolver el problema de Titanic, para el clasificador Naive-Bayes, el codificador CESAMO obtuvo una exactitud de 0.7486. De manera similar, el codificador CESAMMO logró una exactitud de 0.7795. Por otro lado, la técnica de codificación que obtuvo la mejor exactitud para el modelo Naive-Bayes fue el Target Encoder, con un valor de 0.7968. Finalmente, la diferencia absoluta entre la exactitud de CESAMMO y la exactitud del mejor codificador (Target encoder) fue de 0.0173"

Lo anterior aplica para todas las tablas presentadas en esta sección (5.8, 5.9, 5.10, 5.11, 5.12, 5.13).

Tabla 5.8: Resultados del conjunto de datos *Titanic*.

Modelo	**Exactitud CESA-MO**	**Exactitud CESAM-MO**	**C1**	**Exactitud C1**	**Δ(CMMO, C1)**
G. Naive Bayes	0.7486	**0.7795**	Target	0.7968	0.0173
Perceptron mult.	**0.8118**	0.8089	Backward diff.	0.8271	0.0182
Logistic reg.	**0.8174**	0.8146	Backward diff.	0.8215	0.0069
SVM	**0.8315**	0.8174	CESAMO	0.8315	0.0141
XGBoost	**0.8272**	0.823	Backward diff.	0.8283	0.0053

Para el caso del conjunto de datos 'Titanic', se puede observar en la Tabla 5.8 que, para 4 de los 5 modelos, el rendimiento de CESAMO fue mejor que el de CESAMMO. Además, para el modelo Máquina de Soporte Vectorial (SVM), CESAMO superó a todas las demás técnicas de codificación.

Tabla 5.9: Resultados del conjunto de datos *Adult*.

Modelo	**Exactitud CESA-MO**	**Exactitud CESAM-MO**	**C1**	**Exactitud C1**	**Δ(CMMO, C1)**
G. Naive Bayes	**0.806**	0.8035	James Stein	0.8336	0.0301
Perceptron mult.	**0.8522**	0.8521	James Stein	0.8589	0.0068
Logistic reg.	0.8329	**0.8353**	One-Hot	0.8528	0.0175
SVM	**0.8476**	0.8474	Target	0.8547	0.0073
XGBoost	**0.8655**	0.8634	Sum	0.8727	0.0093

En el caso del problema 'Adult' (Tabla 5.9), CESAMO y CESAMMO obtuvieron rendimientos muy similares. Como se puede apreciar, la diferencia se encuentra después del segundo decimal para todas las mediciones

Tabla 5.10: Resultados del conjunto de datos *Credit*.

Modelo	**Exactitud CESA-MO**	**Exactitud CESAM-MO**	**C1**	**Exactitud C1**	**Δ(CMMO, C1)**
G. Naive Bayes	**0.131**	0.1304	James Stein	0.1373	0.0069
Perceptron mult.	0.8778	**0.8814**	Leave one out	0.8914	0.01
Logistic reg.	0.915	**0.916**	Ordinal	0.9161	0.0001
SVM	0.9176	**0.9176**	CESAMMO	0.9176	0.0
XGBoost	0.9174	**0.9176**	Leave one out	0.9176	0.0

En el problema 'Credit' (Tabla 5.10), CESAMMO supera a CESAMO en 3 de los 5 modelos: 1) Red Neuronal Perceptrón Multicapa, 2) Regresión Logística y 3) XGBoost. Nótese que CESAMMO obtuvo el mejor rendimiento para el modelo de SVM.

Tabla 5.11: Resultados del conjunto de datos *Employee*.

Modelo	**Exactitud CESA-MO**	**Exactitud CESAM-MO**	**C1**	**Exactitud C1**	**Δ(CMMO, C1)**
G. Naive Bayes	0.9395	**0.9418**	CESAMMO	0.9418	0.0
Perceptron mult.	0.9409	**0.9422**	Leave one out	0.9431	0.0009
Logistic reg.	0.9418	0.9418	Leave one out	0.9437	0.0019
SVM	0.9418	0.9418	CatBoost	0.944	0.0022
XGBoost	**0.9421**	0.9418	Ordinal	0.9455	0.0037

De manera similar, para el conjunto de datos 'Employee' (Tabla 5.11), CESAMMO obtuvo una mayor exactitud que CESAMO para dos modelos: 1) Gaussian Naive-Bayes y de nuevo 2) Red Neuronal Perceptrón Multicapa. Además, CESAMMO obtuvo el mejor rendimiento entre todas las demás técnicas de codificación para el modelo G. Naive-Bayes.

Tabla 5.12: Resultados del conjunto de datos *Car auction*.

Modelo	Exactitud CESA-MO	Exactitud CESAM-MO	C1	Exactitud C1	Δ(CMMO, C1)
G. Naive Bayes	**0.7045**	0.6951	MEstimate	0.8254	0.1303
Perceptron mult.	**0.8983**	0.8974	Ordinal	0.8989	0.0015
Logistic reg.	**0.8941**	0.8853	CatBoost	0.8952	0.0099
SVM	**0.898**	0.8949	Leave one out	0.8992	0.0043
XGBoost	**0.9**	0.8996	CESAMO	0.9	0.0004

En el conjunto de datos 'Car auction' (Tabla 5.12), CESAMO superó por una pequeña diferencia a CESAMMO para todos los modelos. Es relevante señalar que CESAMO obtuvo el primer lugar entre todos los codificadores para el modelo XGBoost.

Tabla 5.13: Resultados del conjunto de datos *Telco churn*.

Modelo	Exactitud CESA-MO	Exactitud CESAM-MO	C1	Exactitud C1	Δ(CMMO, C1)
G. Naive Bayes	0.5582	**0.7219**	MEstimate	0.7378	0.0159
Perceptron mult.	**0.7886**	0.782	Backward diff.	0.7985	0.0165
Logistic reg.	0.7927	**0.798**	Backward diff.	0.8001	0.0021
SVM	**0.7897**	0.7961	Target	0.7993	0.0032
XGBoost	**0.8016**	0.7991	CESAMO	0.8016	0.0025

Finalmente, para el conjunto de datos 'Telco churn' (Tabla 5.13), CESAMMO superó a CESAMO en dos modelos: 1) Gaussian Naive-Bayes y 2) SVM). De manera similar al problema anterior, CESAMO obtuvo el mejor desempeño de entre todos los codificadores con el modelo de XGBoost.

Es importante destacar que durante la implementación de CESAMMO, se observó un aumento en el costo computacional en comparación con CESAMO, ya que el algoritmo calcula el promedio de al menos 30 evaluaciones para cada conjunto de códigos

candidatos. Este factor debe tenerse en cuenta, especialmente cuando el conjunto de datos contiene múltiples atributos categóricos de alta cardinalidad.

En síntesis, después de evaluar las 60 combinaciones de modelo-codificador se encontró que CESAMO, y su versión multivariable CESAMMO, obtienen desempeños muy similares, siendo incluso, en promedio, ligeramente mejor la versión original.

Si bien nuestra hipótesis inicial era que la consideración multivariable mejoraría el rendimiento de CESAMO, con base en los resultados, se puede rechazar la hipótesis, ya que no hubo mejoras significativas.

Una explicación a esto, es que un atributo categórico no está necesariamente relacionado con todas las demás variables del conjunto de datos, que es lo que busca la versión multivariable. Por lo tanto, es posible que se estén explorando relaciones inexistentes, ya que es más probable que un atributo categórico esté relacionado a una sola variable que con todas. Aunque esta relación univariada puede ser identificada en ambas versiones, CESAMO lo hace de manera más eficiente.

Al hacer un promedio de los rendimientos obtenidos entre todas las posibles combinaciones de modelos y codificadores para los 6 problemas de clasificación resueltos, encontramos que tanto CESAMMO como CESAMO están, en promedio, entre las 5 técnicas con el rendimiento más alto de las 12 técnicas de codificación probadas. Esto se muestra en la Tabla 5.14 (considerando que 1 es el mejor rendimiento y 12 el último).

Tabla 5.14: Ranking de CESAMMO y CESAMO

Problema	**Ranking promedio de CESAMMO (de 12)**	**Ranking promedio de CESAMO (de 12)**
Employee	2.6	3
Auction	3.6	3.4
Credit	3.8	4.8
Telco churn	3.8	4.2
Titanic	7.8	5.2
Adult	9.2	9.2
Promedio total	**5.1**	**4.9**

5.3. Resultados Etapa III: Evaluación estadística de los codificadores

Los resultados expuestos en esta sección se encuentran publicados en el artículo 'Statistical Evaluation of Categorical Encoders for Pattern Preservation in Machine Learning Tasks.' [(51)].

Resumen metodología: en esencia, durante esta etapa se buscó evaluar a las técnicas de codificación empleando un criterio estadístico. Este criterio se basa en el uso de la métrica Error Mínimo (EM) para medir el desempeño de los codificadores. Para realizar la evaluación, fue necesario generar conjuntos de datos sintéticos utilizando funciones polinómicas que simulan las propiedades de los conjuntos de datos reales. Cada conjunto de datos fue codificado utilizando las técnicas a evaluar (incluyendo CESAMO, y descartando CESAMMO). Posteriormente, los conjuntos de datos codificados se procesaron con cinco modelos de aprendizaje automático y se registró el error, definido como $1 -$ exactitud. El proceso se repitió hasta que la distribución de los errores alcanzó normalidad, permitiendo calcular el EM de cada codificador, definido como $\mu - \sqrt{5}\sigma$, proporcionando una medida estadística del rendimiento de los codificadores. Los resultados obtenidos se muestran a continuación.

La evaluación de los codificadores requirió alrededor de 3,000 conjuntos de datos sintéticos para satisfacer el criterio estadístico. Una vez que la Media de la Distribución de Errores (MDE) se volvió Normal, fue posible determinar su media μ_{MDE} y desviación estándar σ_{MDE}. Mediante estos valores, se determinaron la media μ_{DE} y la desviación estándar σ_{DE} de la Distribución de los Errores (DE). Estos dos últimos valores se obtuvieron de la siguiente manera:

$$\mu_{ED} \approx \mu_{MED}, \quad \sigma_{ED} = \sqrt{ss} \cdot \sigma_{MED} \tag{5.1}$$

Donde el tamaño de la muestra ss fue 36. Nótese que estos valores son necesarios para poder determinar el Error Mínimo mediante la Ecuación 4.4

La Tabla 5.15 muestra los valores obtenidos después de calcular μ_{ED} y σ_{ED} para todos los codificadores evaluados.

Tabla 5.15: μ y σ de la Distribución de Errores de los codificadores categóricos evaluados

Codificador	**Error medio**	**Desviación estándar**
CESAMO	0.2994374188	0.0912776249
Hashing	0.3069885601	0.1028621334
Binary	0.3105365472	0.0909518922
One hot	0.3299986654	0.1008621334
Ordinal	0.3315438146	0.1014268199
Count	0.3572900108	0.1106407878

Como se puede ver en la Tabla 5.15, CESAMO fue el codificador que obtuvo el menor error medio entre todas las técnicas de codificación.

Una vez conociendo μ_{DE} y σ_{DE} , se calculó el Error Mínimo para cada técnica de codificación, que de manera resumida es el valor que está a $-\sqrt{5}\sigma_{DE}$ a la izquierda de μ_{DE}. Los resultados se muestran en la Figura 5.8.

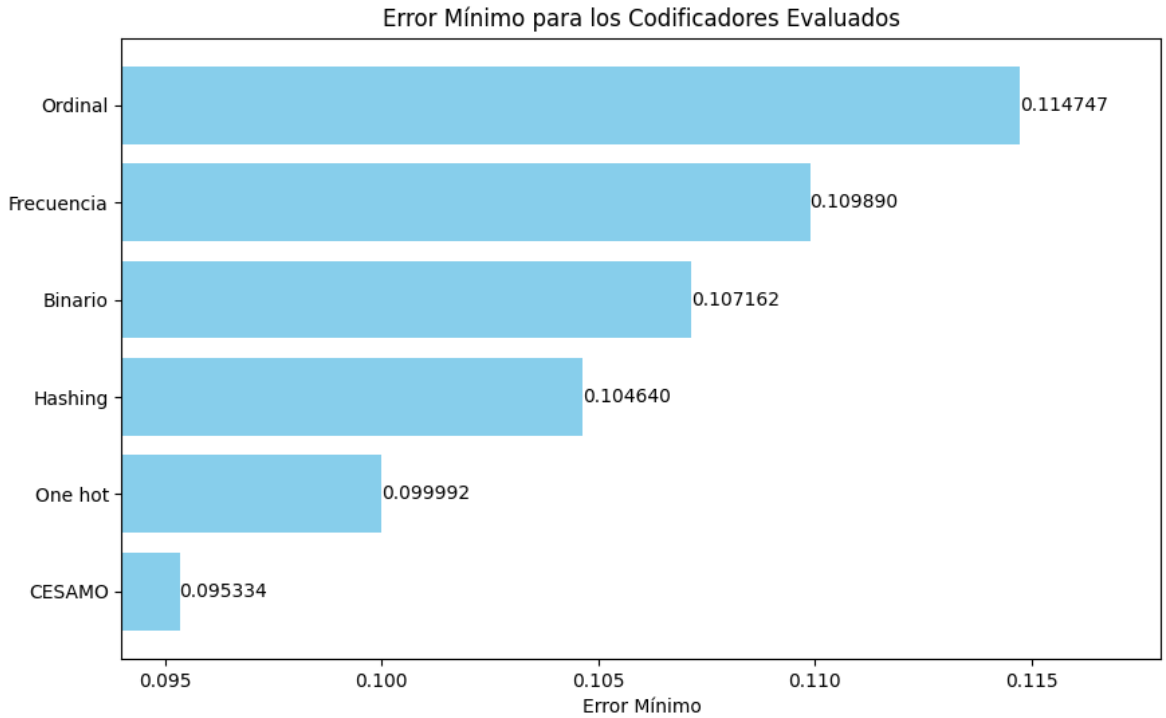

Figura 5.8: Error Mínimo ($\mu_{ED} - \sqrt{5}\sigma_{ED}$) encontrado para los codificadores evaluados

La Figura 5.8 muestra el Error Mínimo obtenido por los codificadores evaluados. Como se puede observar, CESAMO tuvo el menor Error Mínimo entre todos los codificadores, seguido de *One-Hot Encoder*.

Estos hallazgos son consistentes con algunos de los trabajos analizados en el Capítulo Estado del Arte 3. Por ejemplo, en el trabajo (37) también se observó que *One-Hot Encoder* tiene un mejor rendimiento para los modelos basados en Redes Neuronales Perceptron Multicapa. Por otro lado, en el trabajo (38) se encontró de manera similar que *One-Hot Encoder* logró un rendimiento superior con el modelo de regresión logística. Es importante resaltar que en estos estudios se identifican a las combinaciones de modelo-codificador que dan los mejores resultados. En cambio, en nuestro enfoque se calculó el desempeño promedio por codificador, sin mantener la resolución del modelo-codificador. No obstante, los estudios confirman que *One-Hot Encoder* ocupa consistentemente un lugar superior, a pesar de sus conocidas desventajas.

Por otro lado, en la Figura 5.8 también se puede apreciar que el codificador que exhibe el Error Mínimo más alto (lo que se traduce en un peor desempeño), fue el *Ordinal Encoder*, sorprendentemente uno de los codificadores más utilizados en la práctica y literatura. De manera similar, el *Count Encoder* ocupó el segundo lugar más bajo en rendimiento. Estos hallazgos coinciden con los resultados del trabajo (36), donde también se observó un bajo rendimiento en estos codificadores

Capítulo 6

Conclusiones y trabajo futuro

En este capítulo se presenta un breve resumen de las etapas de la investigación junto con las conclusiones finales.

6.1. Conclusiones Etapa I

Durante esta etapa, se cuantificó cómo las diferentes técnicas de codificación pueden afectar a los algoritmos de aprendizaje automático. Este efecto puede variar dependiendo de la naturaleza del atributo categórico (nominal, ordinal) y su cardinalidad.

A partir de los datos del mundo real, encontramos que los modelos de Redes Neuronales Perceptrón Multicapa (MLP) y la Máquina de Soporte Vectorial (SVM) fueron los menos afectados, ya sea de manera positiva o negativa. Una posible explicación a esto, podría ser debido a la capacidad de estos modelos para aprender patrones complejos. Las MLP pueden manejar eficientemente datos con muchas características gracias a su estructura de capas y neuronas, aprendiendo representaciones complejas (59). Por su parte, las SVM, especialmente con *kernels* como el RBF (*Radial Basis Function*), pue-

den operar en espacios de alta dimensionalidad sin incurrir en problemas significativos de sobreajuste (60), lo que también les permite aprender relaciones sofisticadas.

En contraste, el modelo Gaussian Naive-Bayes fue severamente afectado de manera negativa por los codificadores categóricos que añaden atributos para cada instancia. Se encontró que el rendimiento disminuyó alrededor de un 30 %. Esto se debe a que el *One-Hot Encoding* transforma los atributos categóricos en múltiples características binarias, incrementando significativamente la dimensionalidad y creando datos dispersos que no siguen una distribución normal, incumpliendo así el supuesto fundamental del modelo de que las características siguen una distribución gaussiana. Además, las nuevas características binarias están altamente correlacionadas entre sí, ya que provienen del mismo atributo categórico, lo que contradice la suposición de independencia entre las variables, algo que también asume el modelo de Gaussian Naive Bayes (61). Como resultado, el modelo no puede estimar correctamente las probabilidades, lo que conduce a una disminución sustancial en su rendimiento.

También se observó que el codificador *Leave-One-Out* afectó negativamente a los modelos basados en Árboles de Decisión, como *XGBoost* y *Random Forest*. A diferencia de otros codificadores supervisados, *Leave-One-Out* asigna valores distintos a la misma categoría en cada instancia, introduciendo ruido y variabilidad en los datos. Esto afecta la estabilidad de las particiones en los árboles de decisión, ya que pequeñas fluctuaciones en los valores pueden generar divisiones inconsistentes, dificultando la generalización (62). Además, el uso de una media calculada excluyendo la instancia actual introduce una forma sutil de fuga de información, lo que puede hacer que los árboles memoricen patrones espurios en los datos de entrenamiento.

Por otro lado, respecto a los datos sintéticos, un hallazgo relevante fue que se observó que en conjuntos de datos con objetivos no lineales, el modelo de Regresión Logística mostró un mejor rendimiento al aplicar *One Hot Encoder*. De hecho, en todos los casos, el rendimiento de este modelo superó los valores de referencia obtenidos con los datos sintéticos sin atributos categóricos. Una posible explicación es que, en conjuntos de datos con objetivos no lineales, la aplicación de *One Hot Encoding* a las variables categóricas

permitió al modelo de Regresión Logística capturar relaciones no lineales presentes en los datos ([63]). Al transformar las variables categóricas en características binarias, el modelo pudo asignar pesos específicos a cada categoría, introduciendo efectivamente no linealidades. Esto incrementó la capacidad predictiva del modelo, ya que las variables categóricas codificadas aportaron información adicional relevante para la predicción del objetivo.

También se encontró que el modelo *XGBoost* fue el menos afectado por las diferentes técnicas de codificación cuando se emplearon datos sintéticos. La técnica de codificación que obtuvo el mejor rendimiento fue *Catboost Encoder*. En contraste, *Backward Difference encoder* apareció más frecuentemente en combinaciones con el rendimiento más bajo.

En cuanto a la técnica *Ordinal Encoding*, se encontró que obtiene buenos resultados, siempre y cuando la naturaleza del atributo categórico sea ordinal y el orden sea conocido. De lo contrario, si no se aplica de manera correcta, se inducirá un patrón inexistente que afectará negativamente el rendimiento del modelo. En cuanto a los atributos categóricos con alta cardinalidad, se vuelve impráctico aplicar técnicas que incrementan considerablemente el número de columnas (como *One Hot Encoding*).

En conclusión, el uso de datos sintéticos permitió un análisis más rápido y controlado de la efectividad de los codificadores investigados, ya que se conoce la relación entre el conjunto de datos y la variable objetivo. Esto también permitió ahorrar tiempo en la limpieza y preprocesamiento de los datos.

6.2. Conclusiones Etapa II

En esta etapa se esperaba que una versión multivariable de CESAMO tuviera un mejor rendimiento que su versión original. Sin embargo, no se observó una diferencia significativa entre el desempeño de CESAMO y CESAMMO; de hecho, la versión original obtuvo resultados ligeramente superiores.

Una posible explicación es que un atributo categórico no necesariamente está relacionado con todas las demás variables del conjunto de datos, por lo que podría estarse

buscando una relación inexistente. De los resultados obtenidos se puede asumir que es más probable que un atributo categórico esté vinculado a una variable específica dentro del conjunto de datos, en lugar de a todas las demás. Aunque esta relación de una variable es identificada en ambas versiones, CESAMO la encuentra de forma más eficiente.

6.3. Conclusiones Etapa III

En la última etapa, se evaluó el rendimiento de las técnicas de codificación de atributos categóricos utilizando un criterio estadístico. Este criterio consistió en evaluar los codificadores en tantos conjuntos de datos como fuera necesario hasta que la distribución promedio de los errores alcanzara una forma normal.

Para medir y comparar el rendimiento de las técnicas de codificación, se utilizó la métrica del Error Mínimo, definida como el valor situado en $\mu - \sqrt{5}\sigma$ de la Distribución del Error de los codificadores. En promedio, cada combinación de codificador-modelo se evaluó en aproximadamente 3,000 conjuntos de datos sintéticos hasta cumplir con el criterio estadístico.

Los resultados revelaron que CESAMO obtuvo el Error Mínimo más bajo, seguido por el *One-hot Encoder*, mientras que la técnica de *Ordinal Encoding*, ampliamente utilizada, presentó el rendimiento más bajo entre todos los codificadores. Estos hallazgos sugieren que CESAMO superó a los demás codificadores gracias a su capacidad para identificar Códigos que Conservan los Patrones entre el atributo categórico y otros atributos del conjunto de datos.

Por el contrario, como también se encontró en etapas previas, *Ordinal Encoder* introduce patrones inexistentes o información espuria al asignar arbitrariamente valores enteros a las instancias categóricas. Estos resultados concuerdan con los reportados en trabajos previos del estado del arte.

6.4. Contribuciones principales

A continuación se puntualizan las contribuciones de este trabajo de investigación.

- **Cuantificación del impacto de la codificación categórica en modelos de aprendizaje automático**: Se cuantificó cómo las diferentes técnicas de codificación afectan a los modelos de aprendizaje automático, variando el impacto según la naturaleza y cardinalidad del atributo categórico. Se observó que MLP y SVM fueron los modelos menos afectados ante los distintos codificadores, mientras que Gaussian Naive-Bayes sufrió una disminución en su rendimiento con la técnica *One-Hot Encoding*.

- **Identificación de limitaciones en modelos específicos con técnicas de codificación particulares**: Se determinó que el *Leave-One-Out Encoder* afecta negativamente a modelos basados en árboles de decisión, como *XGBoost* y *Random Forest*, debido a la introducción de ruido y sobreajuste al usar valores numéricos distintos para representar a la misma instancia categórica. También se encontró que el *Ordinal Encoder* es efectivo solo cuando la naturaleza ordinal del atributo categórico es conocida y se aplica correctamente; en caso contrario, induce patrones inexistentes que disminuyen el rendimiento del modelo.

- **Confirmación de la utilidad de *One-Hot Encoding* en regresión logística con objetivos no lineales**: Se encontró que en conjuntos de datos con objetivos no lineales, la Regresión Logística mejora significativamente su rendimiento al aplicar *One-Hot Encoding* a los atributos categóricos, permitiéndole capturar relaciones no lineales en los datos.

- **Desarrollo de un marco comparativo con datos sintéticos**: Se diseñó un marco comparativo que, utilizando datos sintéticos, facilitó una evaluación controlada de la efectividad de las técnicas de codificación, ahorrando tiempo en la preparación de datos y permitiendo una exploración más amplia de los codificadores. El generador de datos sintéticos se encuentra disponible en `https://github.com/celestun/polynomial-dataset-generator`.

- **Propuesta y evaluación de una versión multivariable de CESAMO**: se propuso CESAMMO, una versión multivariable de CESAMO, con el fin de mejo-

rar el rendimiento en la codificación. Sin embargo, CESAMO mantuvo un desempeño ligeramente superior, mostrando que la versión original detecta con mayor eficiencia los Códigos que Conservan los Patrones.

- **Implementación de un criterio estadístico y métrica de Error Mínimo para evaluar codificadores**: se evaluó el rendimiento de los codificadores en miles de conjuntos de datos sintéticos mediante un criterio estadístico basado en la normalidad de la distribución de los errores, empleando la métrica de Error Mínimo. Esta métrica estandarizada permitió identificar a CESAMO como el codificador con el menor error. En contraste, *Ordinal Encoder* mostró un rendimiento significativamente inferior. El código del marco comparativo respaldado con un criterio estadístico se encuentra disponible en: `https://github.com/celestun/categorical-encoders-benchmark`.

- **Validación de CESAMO como codificador eficiente**: CESAMO demostró superar a otros codificadores al conservar patrones en los datos. Estos resultados fueron consistentes con trabajos previos en el área, validando la efectividad de CESAMO en comparación con otros métodos de codificación.

6.5. Trabajo futuro

En esta sección se presentan posibles direcciones para ampliar y profundizar el trabajo desarrollado en esta tesis. Las limitaciones encontradas y los hallazgos obtenidos abren diversas oportunidades para futuras investigaciones, que podrían mejorar la comprensión y el rendimiento de las técnicas de codificación de atributos categóricos en modelos de aprendizaje automático.

Una primera línea de investigación podría centrarse en explorar una versión supervisada de CESAMO, tentativamente llamada *Supervised Categorical Encoder*. Aunque CESAMO destaca por su versatilidad en tareas tanto supervisadas como no supervisadas, esta extensión permitiría adaptar la codificación de atributos categóricos es-

pecíficamente a tareas de clasificación supervisada, mejorando la efectividad en contextos donde la información de las etiquetas está disponible.

La modificación propuesta podría consistir en reemplazar el polinomio de grado 11 (Ecuación 3.10), utilizado como función de aproximación en CESAMO, por un modelo de aprendizaje supervisado que capture la relación entre el atributo categórico y la variable objetivo. Entre los algoritmos a considerar para esta tarea, se podrían utilizar redes neuronales de tipo *Multilayer Perceptron* (MLP), capaces de capturar patrones complejos y no lineales en los datos categóricos. Alternativamente, algún algoritmo de *Gradient Boosting*, como *XGBoost*, podría ser una opción, dados sus buenos rendimientos en tareas supervisadas observados en este trabajo.

Otra dirección interesante sería implementar CESAMO en problemas no supervisados, particularmente en *clustering*. Su adaptación a este contexto podría mejorar la agrupación de datos categóricos al conservar patrones sin depender de una variable objetivo. Esto permitiría evaluar si la técnica genera representaciones que preservan la estructura inherente de los datos, facilitando una segmentación más precisa y coherente. Además, su uso en *clustering* reduciría la dependencia de codificaciones estándar como *One-Hot Encoding*, que incrementan la dimensionalidad y dispersan los datos, por lo que se esperaría que CESAMO mejore tanto la eficiencia como la exactitud de los modelos no supervisados.

Bibliografía

[1] I. Goodfellow, *Deep Learning*. MIT Press, 2016.

[2] S. Raschka and V. Mirjalili, *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt publishing ltd, 2019.

[3] A. A. Patel, *Hands-on unsupervised learning using Python: how to build applied machine learning solutions from unlabeled data*. O'Reilly Media, 2019.

[4] J. Han, J. Pei, and H. Tong, *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

[5] T. Hastie, "The elements of statistical learning: data mining, inference, and prediction," 2009.

[6] S. A. Fayaz, S. Jahangeer Sidiq, M. Zaman, and M. A. Butt, "Machine learning: An introduction to reinforcement learning," *Machine Learning and Data Science: Fundamentals and Applications*, pp. 1–22, 2022.

[7] R. S. Sutton, "Reinforcement learning: an introduction," *A Bradford Book*, 2018.

[8] M. Lapan, *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018.

[9] J. Gareth, W. Daniela, H. Trevor, and T. Robert, *An introduction to statistical learning: with applications in R*. Spinger, 2013.

[10] D. Powers and Y. Xie, *Statistical methods for categorical data analysis*. Emerald Group Publishing, 2008.

[11] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. .ºReilly Media, Inc.", 2022.

[12] M. Kuhn and K. Johnson, *Feature engineering and selection: A practical approach for predictive models*. Chapman and Hall/CRC, 2019.

[13] O. Z. Maimon and L. Rokach, *Data mining with decision trees: theory and applications*, vol. 81. World scientific, 2014.

[14] C. Wade and K. Glynn, *Hands-On Gradient Boosting with XGBoost and scikit-learn: Perform accessible machine learning and extreme gradient boosting with Python*. Packt Publishing Ltd, 2020.

[15] M. N. Murty, V. S. Devi, M. N. Murty, and V. S. Devi, "Bayes classifier," *Pattern Recognition: An Algorithmic Approach*, pp. 86–102, 2011.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[17] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[18] R. M. Gray, *Entropy and information theory*. Springer Science & Business Media, 2011.

[19] R. B. Ash, *Information theory*. Courier Corporation, 2012.

[20] M. Li, P. Vitányi, *et al.*, *An introduction to Kolmogorov complexity and its applications*, vol. 3. Springer, 2008.

[21] A. Von Eye and C. C. Clogg, *Categorical variables in developmental research: Methods of analysis*. Elsevier, 1996.

[22] B. Lantz, *Machine learning with R: expert techniques for predictive modeling*. Packt publishing ltd, 2019.

[23] A. Zheng and A. Casari, *Feature engineering for machine learning: principles and techniques for data scientists*. .ºReilly Media, Inc.", 2018.

[24] C. Seger, "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing," 2018.

[25] S. Galli, *Python feature engineering cookbook: over 70 recipes for creating, engineering, and transforming features to build machine learning models*. Packt Publishing Ltd, 2022.

[26] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th annual international conference on machine learning*, pp. 1113–1120, 2009.

[27] D. Micci-Barreca, "A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems," *ACM SIGKDD explorations newsletter*, vol. 3, no. 1, pp. 27–32, 2001.

[28] L. Sasse, E. Nicolaisen-Sobesky, J. Dukart, S. B. Eickhoff, M. Götz, S. Hamdan, V. Komeyer, A. Kulkarni, J. Lahnakoski, B. C. Love, *et al.*, "On leakage in machine learning pipelines," *arXiv preprint arXiv:2311.04179*, 2023.

[29] J. T. Hancock and T. M. Khoshgoftaar, "Catboost for big data: an interdisciplinary review," *Journal of big data*, vol. 7, no. 1, p. 94, 2020.

[30] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[31] A. F. Kuri-Morales, "Minimum database determination and preprocessing for machine learning," in *Innovative Solutions and Applications of Web Services Technology*, pp. 94–131, IGI Global, 2019.

[32] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*, pp. 849–858, Springer, 2000.

[33] A. F. Kuri-Morales, "Categorical encoding with neural networks and genetic algorithms," in *WSEAS Proceedings of the 6th International Conference on Applied Informatics and. Computing Theory*, pp. 167–175, 2015.

[34] A. Kuri-Morales, "Pattern discovery in mixed data bases," in *Pattern Recognition: 10th Mexican Conference, MCPR 2018, Puebla, Mexico, June 27-30, 2018, Proceedings 10*, pp. 178–188, Springer, 2018.

[35] A. Kuri-Morales and A. Cartas-Ayala, "Polynomial multivariate approximation with genetic algorithms," in *Advances in Artificial Intelligence: 27th Canadian Conference on Artificial Intelligence, Canadian AI 2014, Montréal, QC, Canada, May 6-9, 2014. Proceedings 27*, pp. 307–312, Springer, 2014.

[36] F. Pargent, B. Bischl, and J. Thomas, "A benchmark experiment on how to encode categorical features in predictive modeling," *München: Ludwig-Maximilians-Universität München*, 2019.

[37] W. Zhu, R. Qiu, and Y. Fu, "Comparative study on the performance of categorical variable encoders in classification and regression tasks," *arXiv preprint arXiv:2401.09682*, 2024.

[38] F. Matteucci, V. Arzamasov, and K. Böhm, "A benchmark of categorical encoders for binary classification," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[39] D. Seca and J. Mendes-Moreira, "Benchmark of encoders of nominal features for regression," in *World Conference on Information Systems and Technologies*, pp. 146–155, Springer, 2021.

[40] K. Potdar, T. S. Pardawala, and C. D. Pai, "A comparative study of categorical variable encoding techniques for neural network classifiers," *International journal of computer applications*, vol. 175, no. 4, pp. 7–9, 2017.

[41] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *Journal of big data*, vol. 7, no. 1, p. 28, 2020.

[42] E. Valdez-Valenzuela, A. Kuri-Morales, and H. Gomez-Adorno, "Measuring the effect of categorical encoders in machine learning tasks using synthetic data," in *Advances in Computational Intelligence: 20th Mexican International Conference on Artificial Intelligence, MICAI 2021, Mexico City, Mexico, October 25–30, 2021, Proceedings, Part I 20*, pp. 92–107, Springer, 2021.

[43] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

[44] K. Hara, D. Saito, and H. Shouno, "Analysis of function of rectified linear unit used in deep learning," in *2015 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2015.

[45] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[46] W. D. McGinnis, C. Siu, S. Andre, and H. Huang, "Category encoders: a scikit-learn-contrib package of transformers for encoding categorical data," *Journal of Open Source Software*, vol. 3, no. 21, p. 501, 2018.

[47] P. J. M. Ali, R. H. Faraj, E. Koya, P. J. M. Ali, and R. H. Faraj, "Data normalization and standardization: a technical report," *Mach Learn Tech Rep*, vol. 1, no. 1, pp. 1–6, 2014.

[48] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," *Knowledge and information systems*, vol. 34, pp. 483–519, 2013.

[49] E. Valdez-Valenzuela, A. Kuri-Morales, and H. Gomez-Adorno, "Cesammo: Categorical encoding by statistical applied multivariable modeling," in *Mexican International Conference on Artificial Intelligence*, pp. 173–182, Springer, 2022.

[50] R. Rana and R. Singhal, "Chi-square test and its application in hypothesis testing," *Journal of Primary Care Specialties*, vol. 1, no. 1, pp. 69–71, 2015.

[51] E. Valdez-Valenzuela, A. Kuri-Morales, and H. Gomez-Adorno, "Statistical evaluation of categorical encoders for pattern preservation in machine learning tasks.," *International Journal of Combinatorial Optimization Problems & Informatics*, vol. 15, no. 2, 2024.

[52] W. J. Adams, *The life and times of the central limit theorem*, vol. 35. American Mathematical Soc., 2009.

[53] C. Viedma, "Estadística descriptiva e inferencial," *Madrid: ediciones IDT*, 2018.

[54] R. V. Hogg, E. A. Tanis, and D. L. Zimmerman, *Probability and statistical inference*, vol. 993. Macmillan New York, 1977.

[55] B. W. Yap and C. H. Sim, "Comparisons of various types of normality tests," *Journal of Statistical Computation and Simulation*, vol. 81, no. 12, pp. 2141–2155, 2011.

[56] F. W. Scholz and M. A. Stephens, "K-sample anderson–darling tests," *Journal of the American Statistical Association*, vol. 82, no. 399, pp. 918–924, 1987.

[57] M. A. Stephens, "Introduction to kolmogorov (1933) on the empirical determination of a distribution," in *Breakthroughs in statistics: Methodology and distribution*, pp. 93–105, Springer, 1992.

[58] A. F. Kuri-Morales and I. López-Peña, "Normality from monte carlo simulation for statistical validation of computer intensive algorithms," in *Advances in Soft Computing: 15th Mexican International Conference on Artificial Intelligence, MICAI 2016, Cancún, Mexico, October 23–28, 2016, Proceedings, Part II 15*, pp. 3–14, Springer, 2017.

[59] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[60] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[61] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[62] M. Kuhn, "Applied predictive modeling," 2013.

[63] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression*. Springer, 2002.

Apéndice

1. Resultados de los datos sintéticos para las funciones $x^2 + y^2$ y $x^3 + y^3$.

Tabla 1: Resultados para la función $x^2 + y^2$ con solo valores numéricos (valores de referencia)

Algoritmos de aprendizaje automático	**Exactitud para objetivo lineal (%)**	**Exactitud para objetivo no lineal (%)**
Máquina de Vectores de Soporte (SVC)	99.48	96.18
XGBoost (XGB)	98.88	97.43
Red Neuronal (NN)	98.84	94.68
Gaussian Naïve-Bayes (GNB)	95.78	56.09
Regresión Logística (LR)	52.58	52.31

Tabla 2: Resultados para la función $x^2 + y^2$ con x mapeado a valores categóricos y con objetivo lineal

Modelo	**Cod. con mejor rendimiento**	**Mejor exactitud**	**Cod. con peor rendimiento**	**Peor exactitud**	**Dif.**
NN	LOO	90.82	ORD	75.86	14.96
SVC	LOO	90.80	BACK	55.75	35.05
XGB	CAT	90.15	SUM	81.46	8.69
GNB	CAT	85.79	BACK	50.34	35.45
LR	LOO	73.66	ORD	54.45	19.21

Tabla 3: Resultados para la función $x^2 + y^2$ con y mapeado a valores categóricos y con objetivo lineal

Modelo	Cod. con mejor rendimiento	Mejor exactitud	Cod. con peor rendimiento	Peor exactitud	Dif.
NN	LOO	91.08	ORD	76.39	14.69
SVC	CAT	90.95	BACK	55.14	35.81
XGB	ME	90.95	SUM	77.19	13.76
GNB	CAT	85.35	BACK	50.40	34.95
LR	LOO	72.46	ORD	52.26	20.20

Tabla 4: Resultados para la función $x^2 + y^2$ con x mapeado a valores categóricos y con objetivo no lineal

Modelo	Cod. con mejor rendimiento	Mejor exactitud	Cod. con peor rendimiento	Peor exactitud	Dif.
NN	SUM	73.36	BACK	54.21	19.15
SVC	LOO	64.77	BACK	52.50	12.27
XGB	HEL	64.44	CAT	62.51	1.93
LR	CAT	56.17	ORD	51.89	4.29
GNB	JST	55.85	BACK	50.08	5.77

Tabla 5: Resultados para la función $x^2 + y^2$ con y mapeado a valores categóricos y con objetivo no lineal

Modelo	Cod. con mejor rendimiento	Mejor exactitud	Cod. con peor rendimiento	Peor exactitud	Dif.
NN	HEL	74.41	BACK	54.43	19.99
XGB	ORD	66.55	SUM	63.65	2.90
SVC	LOO	65.58	BACK	52.82	12.75
LR	BACK	57.00	ORD	51.61	5.39
GNB	LOO	56.67	BACK	49.86	6.81

Tabla 6: Resultados para la función $x^3 + y^3$ con solo valores numéricos (valores de referencia)

Modelo	**Exactitud para objetivo lineal (%)**	**Exactitud para objetivo no lineal (%)**
Red Neuronal (NN)	99.45	99.53
Máquina de Vectores de Soporte (SVC)	98.98	99.06
XGBoost (XGB)	98.84	99.94
Regresión Logística (LR)	98.27	48.25
Gaussian Naïve-Bayes (GNB)	97.97	94.88

Tabla 7: Resultados para la función $x^3 + y^3$ con x mapeado a valores categóricos y con objetivo lineal

Modelo	**Cod. con mejor rendimiento**	**Mejor exactitud**	**Cod. con peor rendimiento**	**Peor exactitud**	**Dif.**
LR	ONE	91.48	ORD	75.21	16.26
NN	HEL	90.08	ORD	75.39	14.69
SVC	CAT	87.20	BACK	75.44	11.76
GNB	LOO	86.71	BACK	52.89	33.83
XGB	LOO	85.89	SUM	78.39	7.50

Tabla 8: Resultados para la función $x^3 + y^3$ con y mapeado a valores categóricos y con objetivo lineal

Modelo	**Cod. con mejor rendimiento**	**Mejor exactitud**	**Cod. con peor rendimiento**	**Peor exactitud**	**Dif.**
LR	ONE	91.99	ORD	75.59	16.40
NN	SUM	89.76	ORD	75.87	13.89
SVC	LOO	88.13	BACK	76.05	12.08
GNB	LOO	86.80	BACK	52.20	34.60
XGB	LOO	86.63	SUM	79.15	7.48

Tabla 9: Resultados para la función $x^3 + y^3$ con x mapeado a valores categóricos y con objetivo no lineal

Modelo	**Cod. con mejor rendimiento**	**Mejor exactitud**	**Cod. con peor rendimiento**	**Peor exactitud**	**Dif.**
XGB	—	99.95	—	99.95	0.00
NN	CAT	99.59	HEL	83.40	16.19
SVC	CAT	98.49	BACK	53.15	45.34
GNB	ORD	95.99	BACK	51.69	44.30
LR	ONE	63.99	ORD	48.63	15.36

Tabla 10: Resultados para la función $x^3 + y^3$ con y mapeado a valores categóricos y con objetivo no lineal

Modelo	**Cod. con mejor rendimiento**	**Mejor exactitud**	**Cod. con peor rendimiento**	**Peor exactitud**	**Dif.**
XGB	—	99.95	—	99.95	0.00
NN	LOO	99.53	ONE	84.45	15.07
SVC	CAT	98.68	BACK	53.54	45.14
GNB	ORD	95.61	BACK	52.07	43.54
LR	ONE	66.01	ORD	47.84	18.17

2. Conjuntos de datos analizados para generar los datos sintéticos.

Nombre del conjunto de datos	Descripción
Abalone data	Consiste en mediciones biológicas, incluyendo longitud, diámetro, altura y peso de los abalones, junto con el número de anillos que representan su edad.
Car Evaluation Database	Evalúa la aceptabilidad de los automóviles en función de varios atributos. Incluye características como precio, costo de mantenimiento, número de puertas, capacidad de asientos y seguridad.
Hepatitis Domain	Contiene información relacionada con el ámbito médico de la hepatitis. Incluye varios atributos clínicos y de laboratorio, como edad, sexo, síntomas y resultados de pruebas sanguíneas.
breast cancer wisconsin	Está centrado en el diagnóstico de cáncer de mama. Incluye características derivadas de imágenes digitalizadas de biopsias de cáncer de mama, como las propiedades del núcleo celular.
cpu performance	Está relacionado con el rendimiento del hardware de computadoras. Incluye atributos como tiempo de ciclo, tamaño de caché y otras especificaciones técnicas.
Yeast Data Set	Está centrado en el estudio de los patrones de localización de proteínas en la levadura. Incluye atributos relacionados con las secuencias de aminoácidos y otras características de las proteínas de levadura.
Servo Data Set	Se enfoca en el control de un sistema de servomecanismo. Incluye atributos relacionados con las señales de entrada y salida del sistema.
Ecoli Data Set	Está dedicado al estudio de la localización subcelular de proteínas en la bacteria Escherichia coli (E. coli). Incluye atributos relacionados con varias propiedades de las proteínas.
Adult Dataset	También conocido como el conjunto de datos Çensus Income", contiene información demográfica sobre adultos, y la tarea es predecir si una persona gana más de 50,000 dólares por año.
Bank Marketing Dataset	Contiene información relacionada con las campañas de marketing directo de una institución bancaria portuguesa. La tarea es predecir si un cliente suscribirá un depósito a plazo fijo (resultado binario: sí/no).

Tabla 11: Conjuntos de datos utilizados para el análisis.

Todos estos conjuntos de datos están disponibles en: https://archive.ics.uci.edu/